Chp 4 Linear Regression
Chp 5 Logistic regression

Hands-on Machine Learning with R
Boookclub R-Ladies Utrecht and R-Ladies Den Bosch

Martine Jansen

stRt

- Organized by @RLadiesUtrecht and @RLadiesDenBosch
- Meet-ups every 2 weeks on "Hands-On Machine Learning with R"
  by Bradley Boehmke and Brandon Greenwell
- No session recording!But we will publish the slides and notes!
- We use HackMD for making shared notes and for the registry:
  https://hackmd.io/rGu7xw2bRS-Im8Iq7-wvXw
- Please keep mic off during presentation. Nice to have camera on and participate to make the meeting more interactive.
- Questions? Raise hand / write question in HackMD or in chat
- Remember presenters are not necessarily also subject experts ☻
- Remember the R-Ladies code of conduct.
  In summary, please be nice to each other and help us make an **inclusive** meeting! ♥

What did we talk about last time?

- Target engineering: transform outcome variable via `log`/`Box-cox`
- Missingness: informative/random, imputation via estimated statistic/KNN
- Feature filtering: remove (near)-zero variance variables
- Numeric feature engineering: sometimes useful to transformation to reduce skewness, standardization
- Categorical feature engineering: lumping, one-hot / dummy encoding, label encoding
- Dimension reduction: see chp 17-19
- Proper implementation: sequential steps, data leakage, recipes

Part II Supervised Learning
Chp 4 Linear Regression

Approximate (linear) relationship between **continuous** response variable and set of predictor variables

4.1 Prerequisites

Libraries

```
library(dplyr)     # for data manipulation
library(ggplot2)   # for graphics

library(caret)     # for cross-validation, etc.
library(rsample)   # you have to scroll back in the book to detect
                   # necessary for initial_split
library(vip)       # variable importance
#library(pdp)      # is used in section on varible importance
```

Code for the data, from previous chps

```
ames <- AmesHousing::make_ames()

set.seed(123)
split <- initial_split(ames, prop = 0.7,
                       strata = "Sale_Price")
ames_train  <- training(split)
ames_test   <- testing(split)
```

4.2 Simple linear regression

If Y and X are (approx) linearly related then:

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i, \text{ for } i = 1, \ldots, n, \text{ and } \epsilon_i \sim N(0, \sigma^2)$$

β_0: intercept, average response when X = 0

β_1: slope, increase in average response per 1 unit increase in X

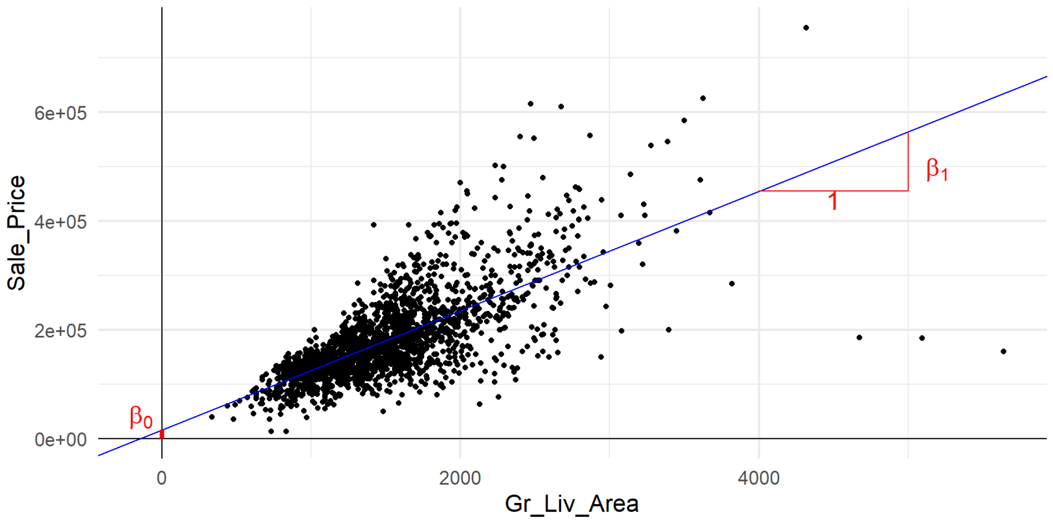
Using least squares regression, coefficients can be calculated with `lm`:

```
model1 <- lm(Sale_Price ~ Gr_Liv_Ar
             data = ames_train)
model1$coef
```

```
(Intercept) Gr_Liv_Area
 15938.1733    109.6675
```

```
sigma(model1)
```

```
[1] 56787.94
```



Inference

- Point estimates for β_0, β_1 and σ not that interesting
- Need to know how much they vary
- When these assumptions are met:
  - independent obs
  - random error mean zero, constant variance
  - random error normally distributed
  $100(1-\alpha)\%$ confidence interval: $\beta_j \pm t_{1-\alpha/2,n-p}\widehat{SE}_{\beta_j}$

```r
confint(model1, level = 0.95)
```

```
                2.5 %     97.5 %
(Intercept) 8384.213 23492.1336
Gr_Liv_Area  104.920   114.4149
```

```r
summary(model1)
```

```

Call:
lm(formula = Sale_Price ~ Gr_Liv_Area, data = ames_train)

Residuals:
    Min      1Q  Median      3Q     Max 
-474682  -30794   -1678   23353  328183 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 15938.173   3851.853   4.138 3.65e-05 ***
Gr_Liv_Area   109.667      2.421  45.303  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 56790 on 2047 degrees of freedom
Multiple R-squared:  0.5007,	Adjusted R-squared:  0.5004 
F-statistic:  2052 on 1 and 2047 DF,  p-value: < 2.2e-16
```

4.3 Multiple linear regression

- More main effects:
  `model2 <- lm(Sale_Price ~ Gr_Liv_Area + Year_Built, data = ames_train)`
- Add an interaction effect with `:`:
  `model2a <- lm(Sale_Price ~ Gr_Liv_Area + Year_Built + Gr_Liv_Area:Year_Built, data = ames_train)`
- Add all the features in the set as main effects:
  `model3 <- lm(Sale_Price ~ ., data = ames_train)`
- The analyst decides on having interaction effects in linear regression
- When interaction effect in model, have also comprising terms in model

4.4 Assessing model accuracy

For this example, “best”model: lowest RMSE via cross-validation

```r
set.seed(123)  # for reproducibility
(cv_model1 <- train(
  form = Sale_Price ~ Gr_Liv_Area,
  data = ames_train,
  method = "lm",
  trControl = trainControl(method = "cv", number = 10)
))
```

```
Linear Regression

2049 samples
   1 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1843, 1844, 1844, 1844, 1844, 1844, ...
Resampling results:

  RMSE      Rsquared  MAE
  56644.76  0.510273  38851.99

Tuning parameter 'intercept' was held constant at a value of TRUE
```

The (averaged) RMSE for the 3 main effect models:

```
cv_model1$results$RMSE
cv_model2$results$RMSE
cv_model3$results$RMSE
```

```
[1] 56644.76

[1] 46865.68

[1] 41691.74
```

Interpret the cv result as:
When applied to unseen data, the predictions model 3 makes are, on average, about 41691.74 off from the actual sale price.

Looking at adjusted R^2, as I got taught:

```
summary(model1)$adj.r.squared
summary(model2)$adj.r.squared
summary(model3)$adj.r.squared
```

```
[1] 0.5004063

[1] 0.6567094

[1] 0.9339612
```

Model 3 explains 94% of the variance in sale price

4.5 Model concerns

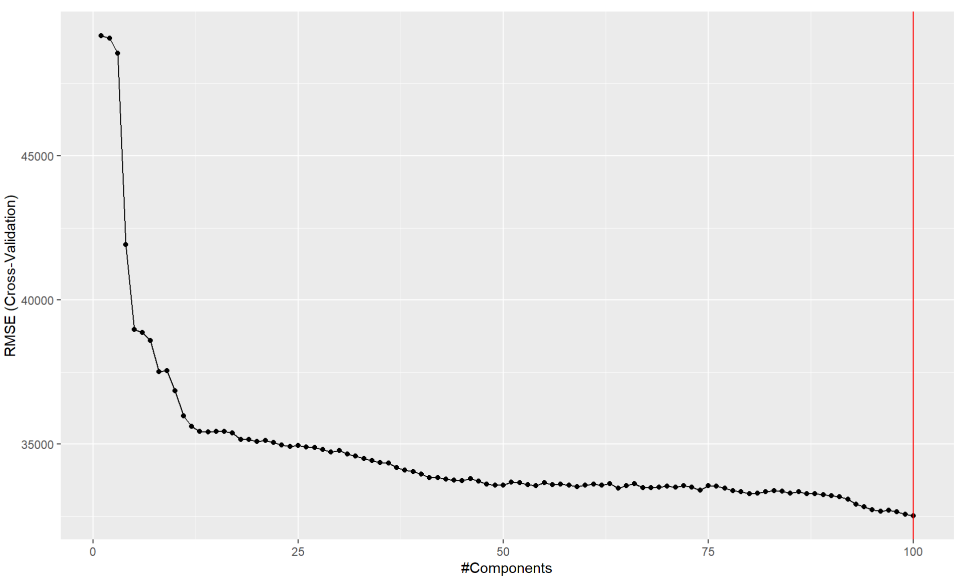
Be sure the assumptions hold:

- Linear relationship, if not transform
- Constant variance among residuals (homoscedasticity)
- No autocorrelation, errors are independent and uncorrelated
- More observations than predictors, if not try regularized regression
- No or little multicollinearity, if not difficult to separate out individual effects variables

4.6 Principal component regression

Address multicollinearity for instance by using Principal Components as predictors

```
set.seed(123)
cv_model_pcr <- train(
  Sale_Price ~ .,
  data = ames_train,
  method = "pcr",
  trControl = trainControl(method = "cv
                           number = 10)
  preProcess = c("zv", "center", "scale
  tuneLength = 100)

bestTune <- cv_model_pcr$bestTune[1,1]

ggplot(cv_model_pcr) +
  geom_vline(xintercept = bestTune,
             color = "red")
```

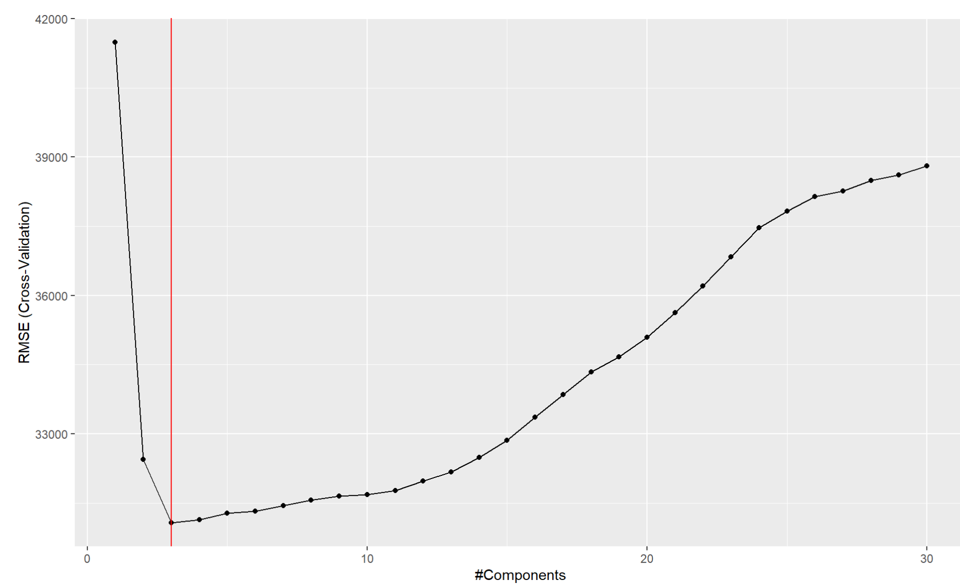


Question I have: - Why useful? It brings RMSE down, but do we get insight in importance of predictors? Different from regular regression? - I thought there are max ncol(data) PC's

Partial least squares

Supervised dimension reduction procedure:
- that finds new features - that not only captures most information in original features, - but **also are related to the response** - PLS places highest weight on variables most strongly related to response

```r
set.seed(123)
cv_model_pls <- train(
  Sale_Price ~ .,
  data = ames_train,
  method = "pls",
  trControl = trainControl(method = "cv
                           number = 10)
  preProcess = c("zv", "center", "scale
  tuneLength = 30
)

bestTune <- cv_model_pls$bestTune[1,1]

ggplot(cv_model_pls) +
  geom_vline(xintercept = bestTune,
             color = "red")
```



Added, how to see this best PLCmodel

```r
the_best_pls <- cv_model_pls$finalModel
the_best_pls$coefficients
```

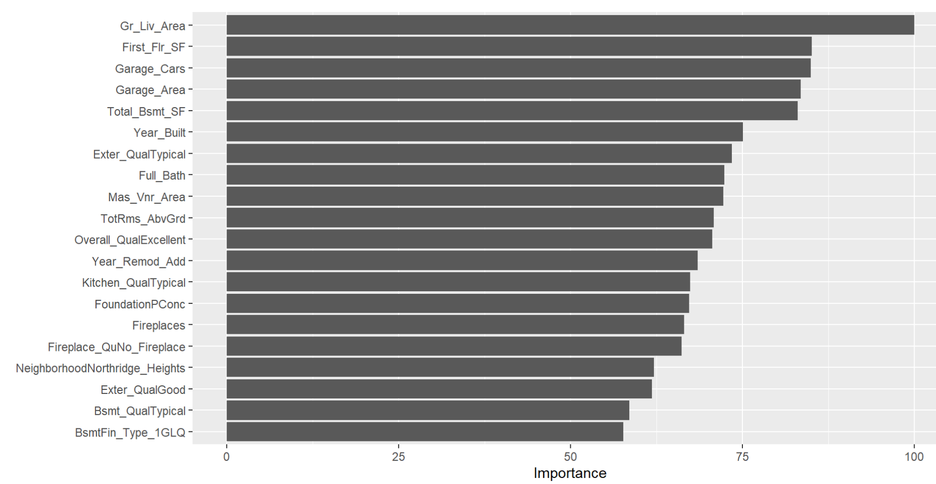
```
, , 1 comps

                                                 .outcome
MS_SubClassOne_Story_1945_and_Older           -1148.4437622
MS_SubClassOne_Story_with_Finished_Attic_All_Ages  -147.5518230
MS_SubClassOne_and_Half_Story_Unfinished_All_Ages  -337.5525120
MS_SubClassOne_and_Half_Story_Finished_All_Ages    -865.0621444
MS_SubClassTwo_Story_1946_and_Newer            1790.3091140
MS_SubClassTwo_Story_1945_and_Older            -294.2140110
MS_SubClassTwo_and_Half_Story_All_Ages          142.9321733
MS_SubClassSplit_or_Multilevel                 -116.6483406
MS_SubClassSplit_Foyer                         -233.9429173
MS_SubClassDuplex_All_Styles_and_Ages          -499.7477346
MS_SubClassOne_Story_PUD_1946_and_Newer         474.0964976
MS_SubClassTwo_Story_PUD_1946_and_Newer        -529.4888895
MS_SubClassPUD_Multilevel_Split_Level_Foyer    -339.6653643
MS_SubClassTwo_Family_conversion_All_Styles_and_Ages  -463.2790399
MS_ZoningResidential_High_Density              -251.1720053
```

4.8 Feature interpretation

- Variable importance: identify variables most influential in model
- LR: often absolute value t-statistic for each parameter
- Difficult when having interactions and transformations
- PLS: contribution coefficients weighted proportionally to reduction RSS

Calculate VIP in PLS
(100 is most important):

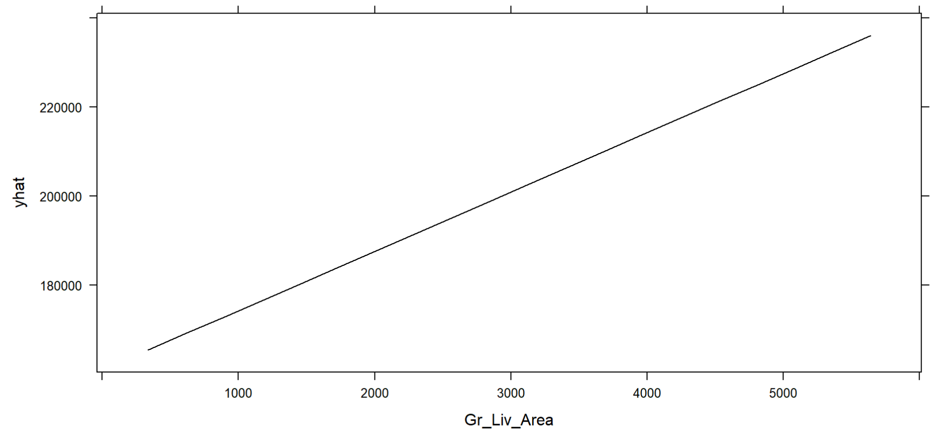
```
vip(cv_model_pls,
    num_features = 20,
    method = "model")
```



PDP - partial dependence plots

- Plot change in average predicted value as specified feature(s) vary over their marginal distribution
- How fixed change in a predictor relates to fixed linear change in outcome, while taking into account average effect of all other features in model
- More useful in case of non-linear relationships (chp 16)

```r
# This is NOT a ggplot!
pdp::partial(cv_model_pls,
             "Gr_Liv_Area",
             grid.resolution = 20,
             plot = TRUE)
```



Part II Supervised Learning
Chp 5 Logistic Regression

Approximate the relationship between a **binary** response variable and a set of predictor variables

5.1 Prerequisites

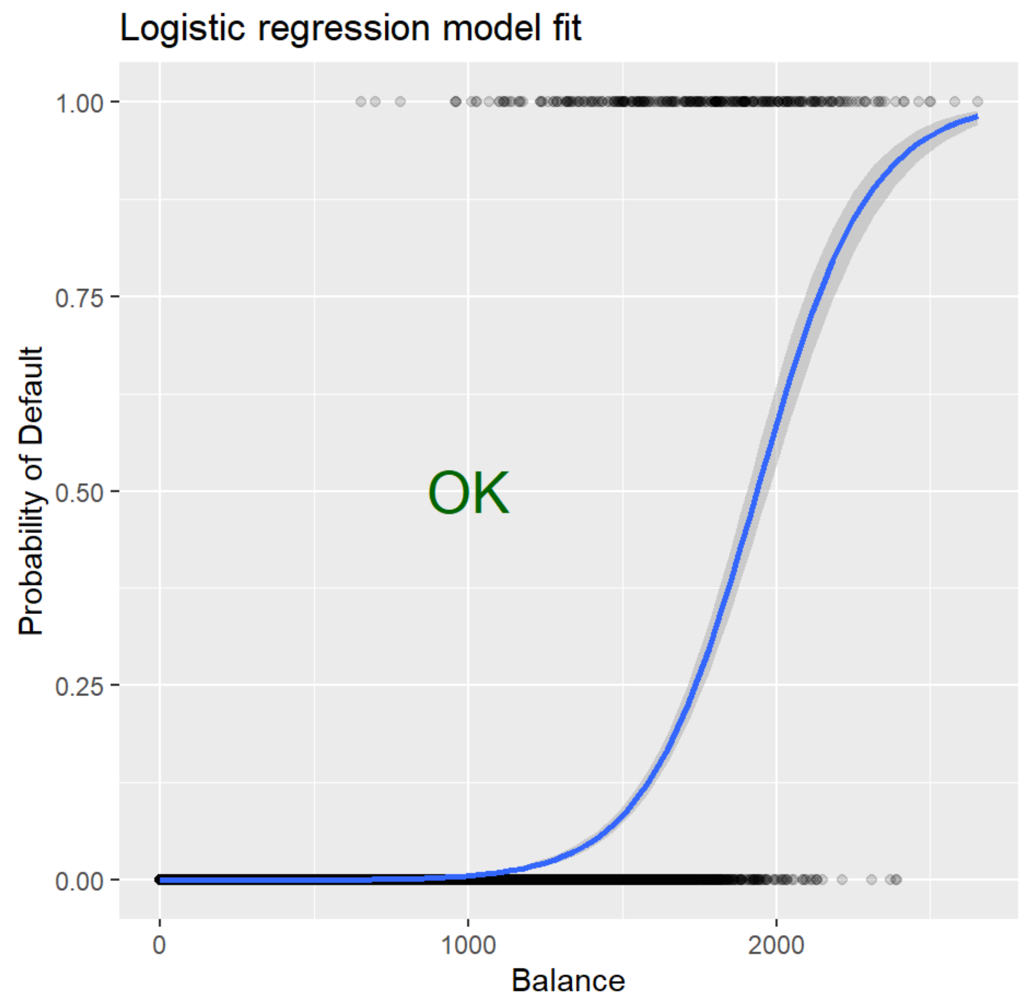
Libraries

```
library(dplyr)     # for data manipulation
library(ggplot2)   # for graphics
library(caret)     # for cross-validation, etc.
library(rsample)   # necessary for initial_split
library(vip)       # variable importance
# library(modeldata)
# library(broom)
# library(ROCR)
```

Code for the data, from previous chps

```
# attrition <- rsample::attrition # line in book chp1  no longer works

# data are moved into the `modeldata` package
df <- modeldata::attrition %>%
  # make all factors unordered
  mutate_if(is.ordered, factor, ordered = FALSE)

set.seed(123)  # for reproducibility
churn_split <- initial_split(df, prop = .7, strata = "Attrition")
```

5.2 Why logistic regression



The formula of a sigmoid function looks complicated:

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

Look at odds:

$$\frac{p(X)}{1 - p(X)} = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \Big/ \frac{1}{1 + e^{\beta_0 + \beta_1 X}} = e^{\beta_0 + \beta_1 X}$$

And then take log, and call that logit (the log of the odds):

$$log\left(\frac{p(X)}{1 - p(X)}\right) = log\left(e^{\beta_0 + \beta_1 X}\right) = \beta_0 + \beta_1 X$$

5.3 Simple logistic regression

Models are calculated using Maximum Likelihood

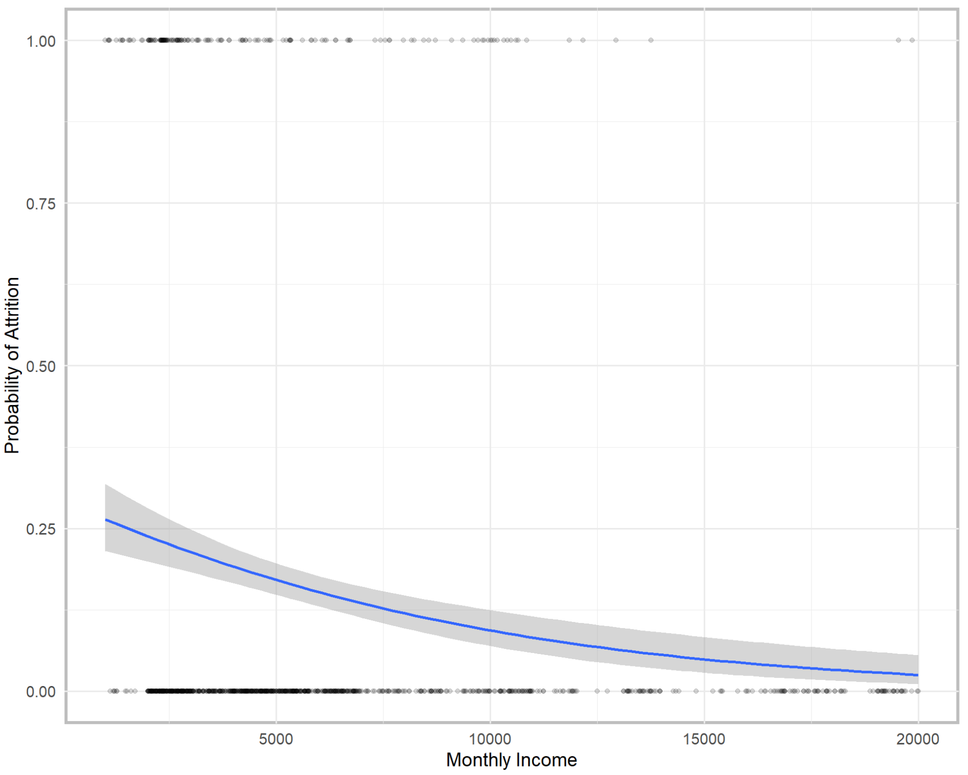
```r
model1 <- glm(Attrition ~ MonthlyIncome,
              family = "binomial",
              data = churn_train)
```

```r
broom::tidy(model1)[,1:2] %>%
  knitr::kable(booktabs = TRUE)
```

term	estimate
(Intercept)	-0.8860896
MonthlyIncome	-0.0001386

Increase of 1 unit in MonthlyIncome,

- logit of attrition 0.000139 less
- odds of attrition multiply by exp(-0.000139) = 0.99986
- hence odds smaller, hence probability smaller



Confidence interval for coefficient

```r
tidy(model1)
```

```
# A tibble: 2 x 5
  term          estimate std.error statistic      p.value
  <chr>            <dbl>     <dbl>     <dbl>        <dbl>
1 (Intercept)  -0.886    0.157         -5.64 0.0000000174
2 MonthlyIncome -0.000139 0.0000272     -5.10 0.000000344
```

```r
# for the logit coefficients:
confint(model1)
```

```
                      2.5 %        97.5 %
(Intercept)   -1.1932606571 -5.761048e-01
MonthlyIncome -0.0001948723 -8.803311e-05
```

```r
# for the odds coefficients:
exp(confint(model1))
```

```
                  2.5 %    97.5 %
(Intercept)   0.3032309 0.5620835
MonthlyIncome 0.9998051 0.9999120
```

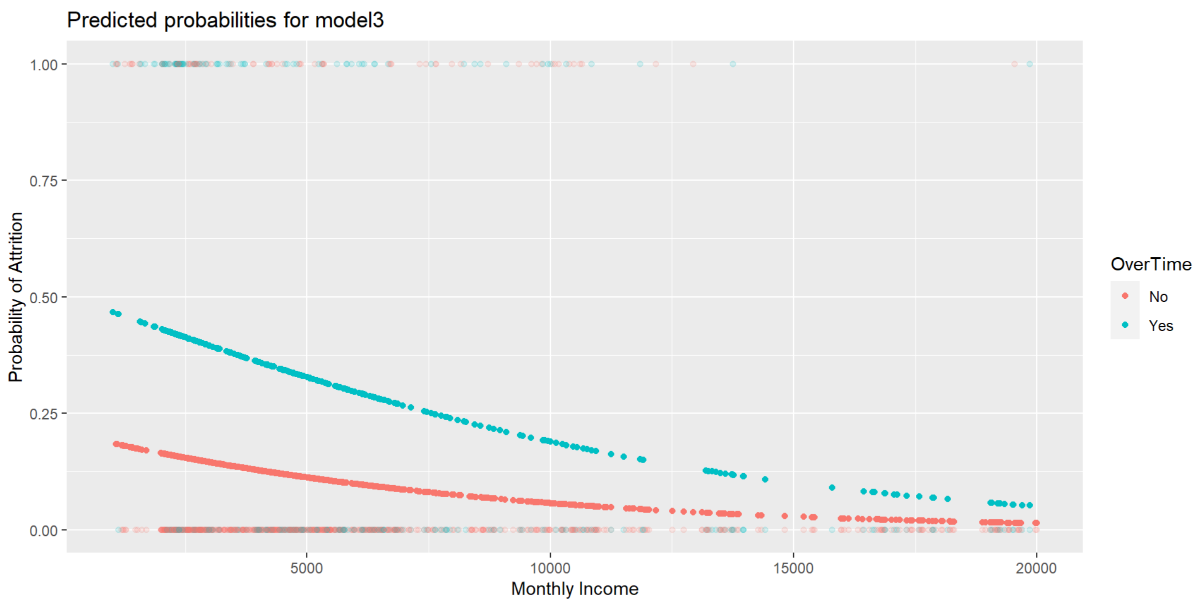
5.4 Multiple logistic regression

Explaining attrition from MonthlyIncome and Overtime:

```r
model3 <- glm(
  Attrition ~ MonthlyIncome + OverTime,
  family = "binomial",
  data = churn_train
  )

broom::tidy(model3)
```

```
# A tibble: 3 x 5
  term          estimate std.error statistic  p.value
  <chr>            <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   -1.33    0.177         -7.54 4.74e-14
2 MonthlyIncome -0.000147 0.0000280     -5.27 1.38e- 7
3 OverTimeYes    1.35    0.180          7.50 6.59e-14
```

```r
churn_train3 <- # different from book:
  # adds column "pred" to data
  # with probs according to model 3
  modelr::add_predictions(churn_train, model = model3, type = "response") %>%
  mutate(prob = ifelse(Attrition == "Yes", 1, 0))

# also different from book
ggplot(churn_train3,
       aes(x = MonthlyIncome, color = OverTime)) +
  geom_point(aes(y = prob), alpha = .15) +        # observations
  geom_point(aes(y = pred)) +                     # predictions
  labs(title = "Predicted probabilities for model3",
       x = "Monthly Income",
       y = "Probability of Attrition")
```



5.5 Assessing model accuracy - how well models predict

Attrition ~ MonthlyIncome

```r
set.seed(123)
cv_model1 <- train(
  Attrition ~ MonthlyIncome,
  data = churn_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv
                           number = 10)

pred_class1 <- predict(cv_model1,
                       churn_train)

confusionMatrix(
  data = relevel(pred_class1,
                 ref = "Yes"),
  reference =
    relevel(churn_train$Attrition,
            ref = "Yes")
)
```

Attrition ~ .

```r
set.seed(123)
cv_model3 <- train(
  Attrition ~ .,
  data = churn_train,
  method = "glm",
  family = "binomial",
  trControl = trainControl(method = "cv
                           number = 10)

pred_class3 <- predict(cv_model3,
                       churn_train)

confusionMatrix(
  data = relevel(pred_class3,
                 ref = "Yes"),
  reference =
    relevel(churn_train$Attrition,
            ref = "Yes")
)
```

`Attrition ~ MonthlyIncome`

```
Confusion Matrix and Statistics

          Reference
Prediction Yes  No
       Yes   0   0
       No  165 863

               Accuracy : 0.8395
                 95% CI : (0.8156, 0.8614)
    No Information Rate : 0.8395
    P-Value [Acc > NIR] : 0.5208

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.8395
             Prevalence : 0.1605
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000
```

`Attrition ~ .`

```
Confusion Matrix and Statistics

          Reference
Prediction Yes  No
       Yes  83  20
       No   82 843

               Accuracy : 0.9008
                 95% CI : (0.8809, 0.9184)
    No Information Rate : 0.8395
    P-Value [Acc > NIR] : 8.982e-09

                  Kappa : 0.5658

 Mcnemar's Test P-Value : 1.542e-09

            Sensitivity : 0.50303
            Specificity : 0.97683
         Pos Pred Value : 0.80583
         Neg Pred Value : 0.91135
             Prevalence : 0.16051
         Detection Rate : 0.08074
   Detection Prevalence : 0.10019
      Balanced Accuracy : 0.73993
```

`No Information Rate : 0.8395`: Predict most common outcome (“No”) for all, still accuracy 83.9%.

`Accuracy`: P(pred = actual), (TP+TN)/(TP+FP+TN+FN)

`Sensitivity` (recall): P(pred = “yes”| actual = “yes”), TP / (TP + FN)

`Specificity`: P(pred = “no”| actual = “no”), TN / (TN + FP)

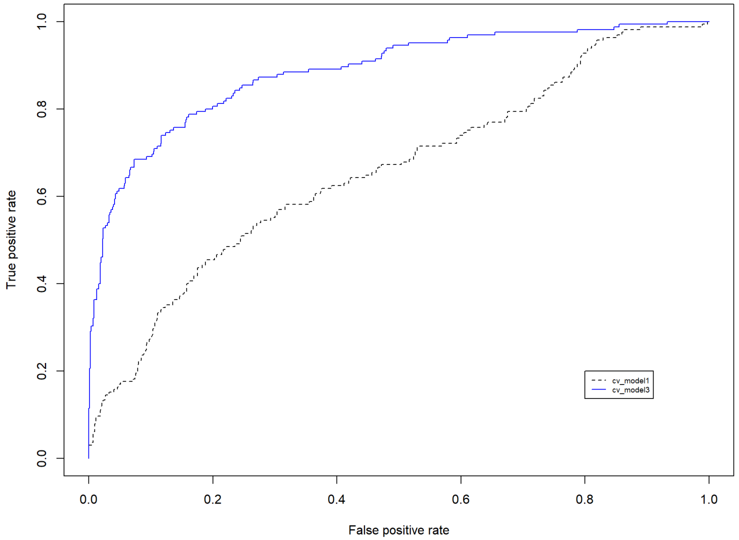
`Pos Pred Value` (precision): P(actual = “yes”| pred = “yes”), TP / (TP + FP)

`Neg Pred Value`: P(actual = “no”| pred = “no”), TN / (TN + FN)

`Prevalence`: (TP+FN)/(TP+FN+FP+FN)

ROC curve

```r
library(ROCR)

m1_prob <- predict(cv_model1,
          churn_train, type = "prob")$Yes
m3_prob <- predict(cv_model3,
          churn_train, type = "prob")$Yes

# Compute AUC metrics for models
perf1 <- prediction(m1_prob,
                    churn_train$Attrition) %>%
  performance(measure = "tpr",
              x.measure = "fpr")
perf2 <- prediction(m3_prob,
                    churn_train$Attrition) %>%
  performance(measure = "tpr",
              x.measure = "fpr")

plot(perf1, col = "black", lty = 2)
plot(perf2, add = TRUE, col = "blue")
legend(0.8, 0.2, legend = c("cv_model1", "cv_mode
       col = c("black", "blue"), lty = 2:1, cex
```



Other options for ROC curves: https://rviews.rstudio.com/20 r-packages-for-roc-curves/

5.6 Model concerns

- Also important to check adequacy
- Concept of residual is difficult
- Some literature referals

5.7 Feature interpretation

```
vip(cv_model3, num_features = 20)
```

5.8 Final thoughts

- Logistic regression suffers also from the many assumptions (i.e. linear relationship of the coefficient, multicollinearity)
- Often more than two classes to predict (multinomial classification)
- Future chapters discuss more advanced algorithms for binary and multinomial classification