Lab 04B: Week 12 (Solutions)

Contents

The **specific aims** of this lab are:

- estimate logistic regression models
- identify which variables are significant and perform model selection
- interpret the coefficients from a logistic regression model
- use decision trees and random forests for classification
- evaluate out-of-sample performance using cross validation
- use estimated prediction/classification models to predict the outcomes for new observations

The unit **learning outcomes** addressed are:

- LO1 Formulate domain/context specific questions and identify appropriate statistical analysis.
- LO3 Construct, interpret and compare numerical and graphical summaries of different data types including large and/or complex data sets.
- LO7 Perform statistical machine learning using a given classifier, and create a cross-validation scheme to calculate the prediction accuracy.

1 Questions

1.1 Who has diabetes?

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases (Johannes 1988). The objective is to predict whether or not a patient has diabetes based on certain clinical measurements. The larger database of patients has been subset such that all observations here are females at least 21 years old of Pima Indian heritage.

The data sets consists of several medical predictor variables and one target variable, `y` which equals 1 if an individual is diabetic and 0 otherwise. Predictor variables includes the number of pregnancies the patient has had (`npreg`), their BMI, insulin level (`serum`), age, triceps skin fold thickness `skin`, diastolic blood pressure (`bp`), plasma glucose concentration (`glu`) and diabetes pedigree function (`ped`).

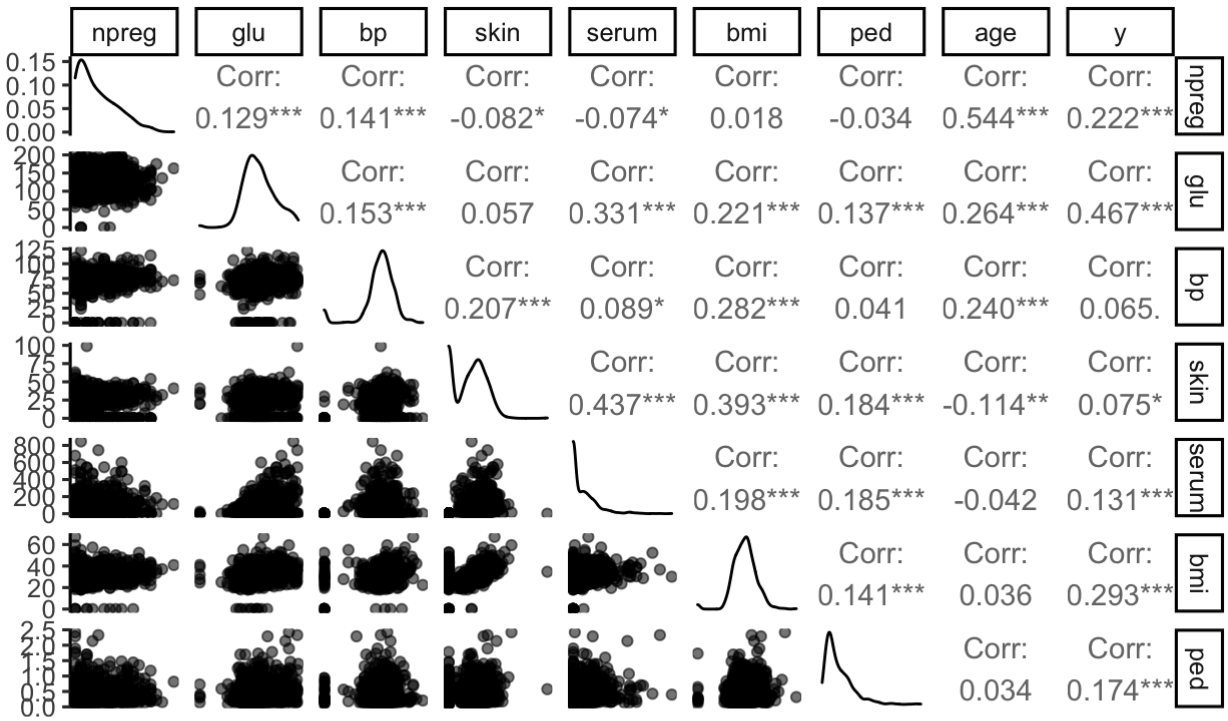
It is available from various places, including Kaggle and the **reglogit** R package and I've uploaded a copy to the GitHub server.

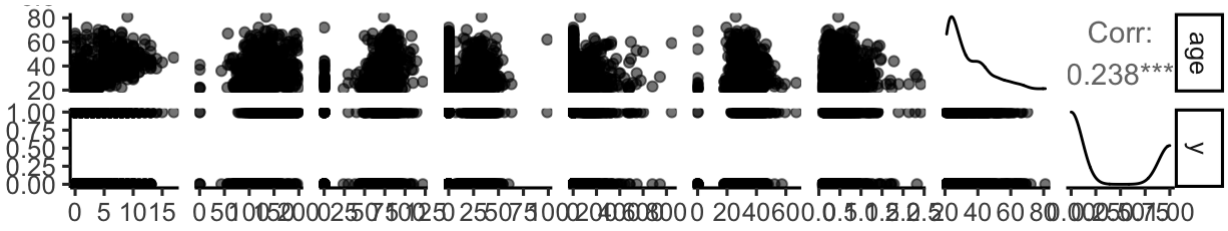
```
library(tidyverse)
pima = readr::read_csv("https://raw.githubusercontent.com/DATA2002/data/master/pima.csv")
glimpse(pima)
```

```
Rows: 768
Columns: 9
$ npreg <dbl> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, 5, 7, 0, …
$ glu   <dbl> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125, 110, 16…
$ bp    <dbl> 72, 66, 64, 66, 40, 74, 50, 0, 70, 96, 92, 74, 80, 60,…
$ skin  <dbl> 35, 29, 0, 23, 35, 0, 32, 0, 45, 0, 0, 0, 0, 23, 19, 0…
$ serum <dbl> 0, 0, 0, 94, 168, 0, 88, 0, 543, 0, 0, 0, 0, 846, 175,…
$ bmi   <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.3, 30.5, …
$ ped   <dbl> 0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.248, 0.134…
$ age   <dbl> 50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 34, 57, 59…
$ y     <dbl> 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, …
```

1. Visualise the data and look for any obvious relationships or patterns in the data. Perform any data cleaning as appropriate.

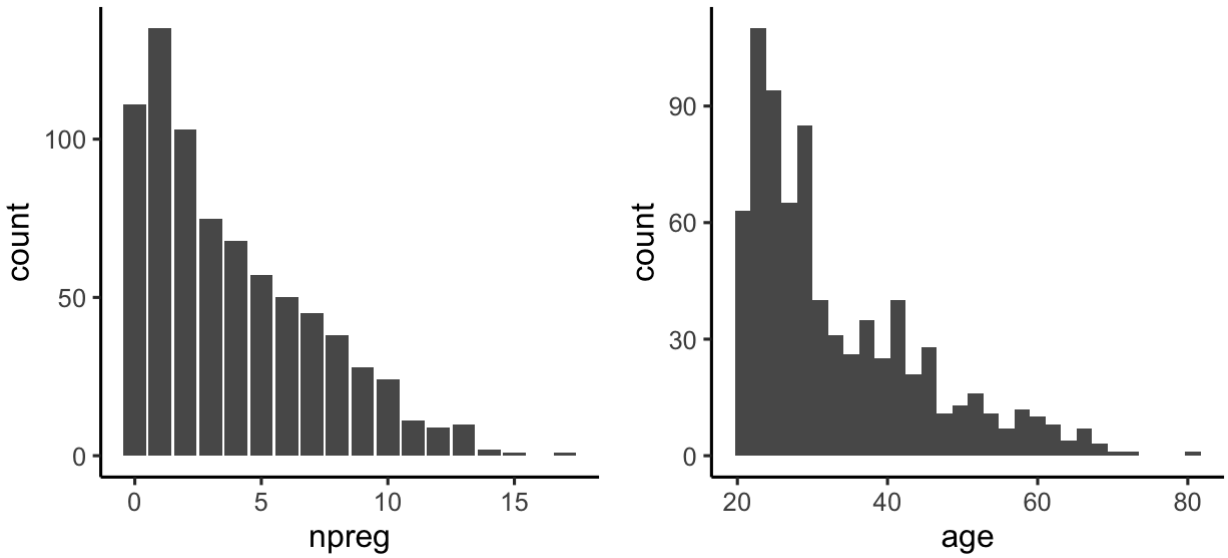
```
GGally::ggpairs(pima, aes(alpha = 0.05))
```



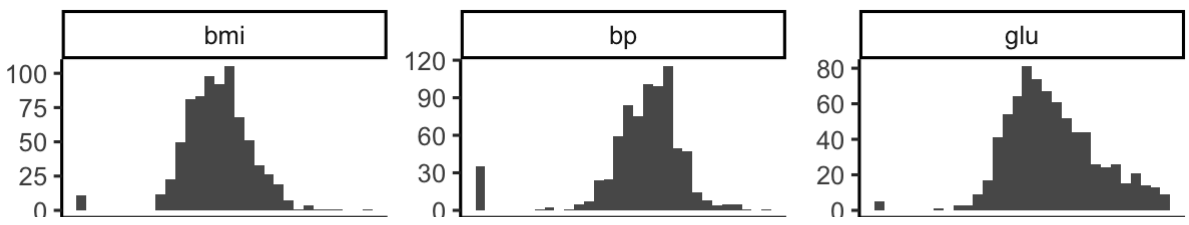


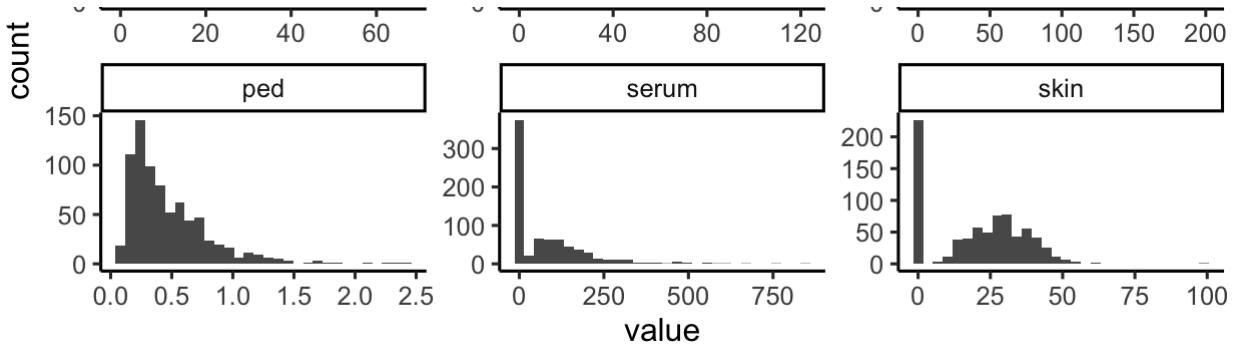
Looks like there are some unwelcome zeros in the data. Let's go variable by variable.

```
p1 = pima %>%
    ggplot(aes(x = npreg)) + geom_bar()
p2 = pima %>%
    ggplot(aes(x = age)) + geom_histogram()
gridExtra::grid.arrange(p1, p2, ncol = 2)
```




```
p2dat = pima %>%
    dplyr::select(glu:ped) %>%
    gather()
ggplot(p2dat) + aes(x = value) + geom_histogram() + facet_wrap(~key, scales = "free")
```

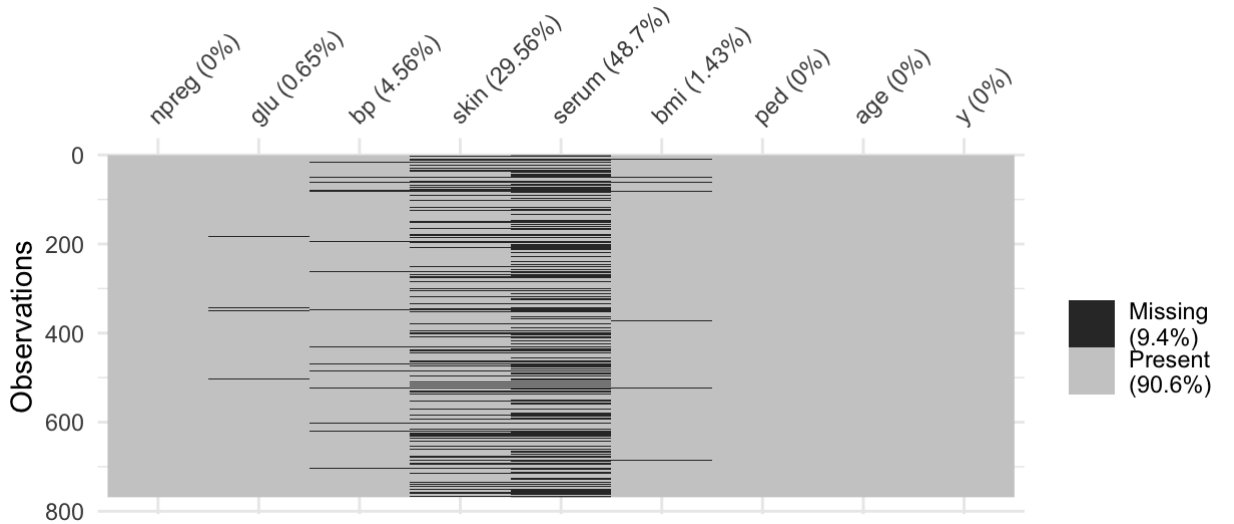




The zeros in `bmi`, `bp`, `glu`, `serum` and `skin` don't make sense. We could convert them to `NA` and drop them from the analysis:

```
pima_clean = pima %>%
  dplyr::mutate(
    dplyr::across(c(bmi, bp, glu, serum, skin),
                  .fns = ~ dplyr::na_if(., 0))
  )
```

```
visdat::vis_miss(pima_clean) +
  theme(legend.position = "right")
```




```
pima_red = pima_clean %>% drop_na()
nrow(pima_red)
```

```
[1] 392
```

We can see that the serum variable is particularly troublesome with almost half of the observations missing. The `skin` variable also has a substantial proportion of missing values.

If we drop any observations that have a missing value, this leaves us with only 392 from the original 768 data (around half of the observations had at least one missing value).

Another alternative is to **impute** the missing values. A simple way to do this is to replace any `NA` values with the mean of that variable.

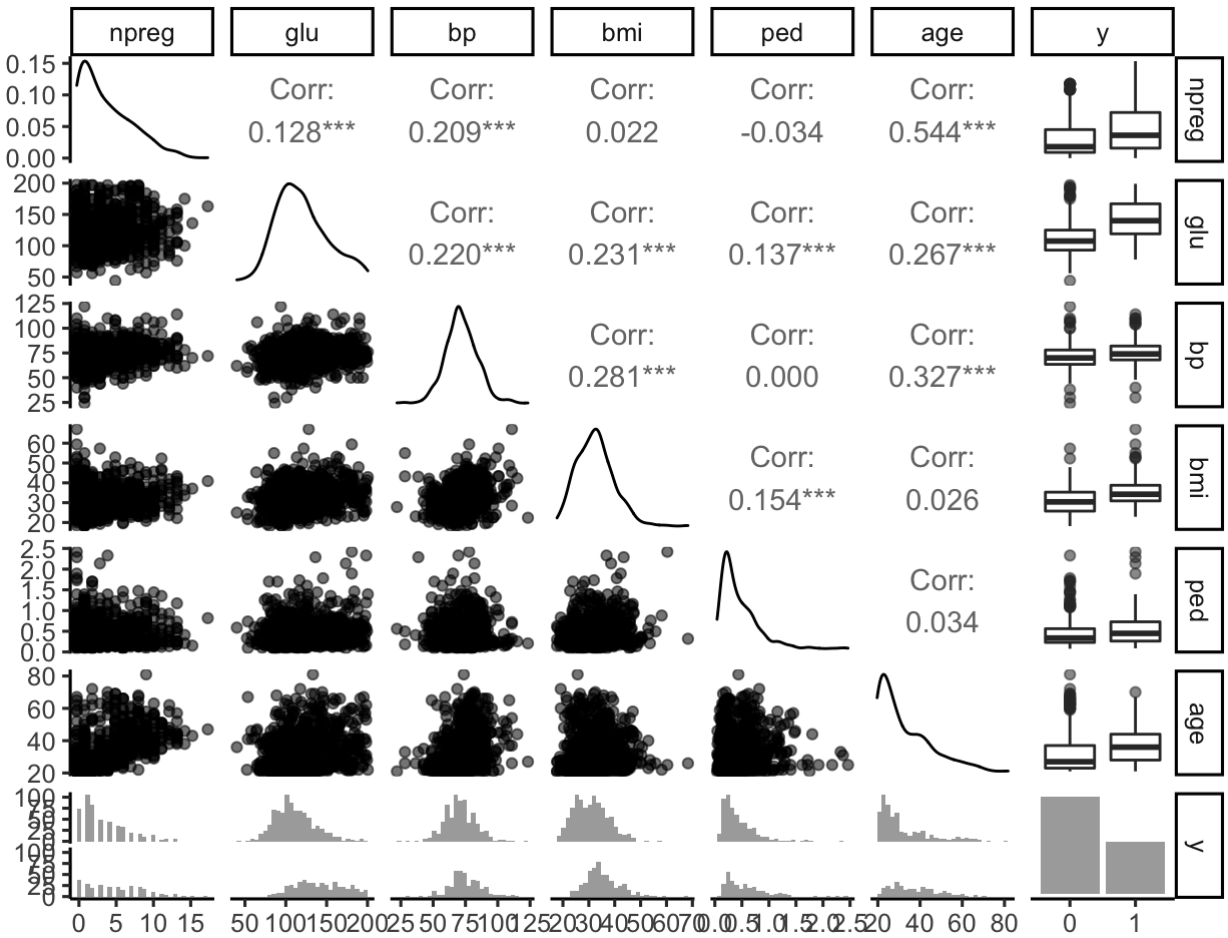
```
pima_impute = pima %>%
  mutate(
    across(c(bmi, bp, glu, serum, skin),
           .fns = ~ ifelse(. == 0, mean(., na.rm= TRUE), .))
  )
```

If we proceeded with the `pima_impute` data, we would need to be cautious about reading too much into the results about the serum and skin variables.

Equally, proceeding with the reduced data set might mean we're excluding an important subset of people, those who don't have access to medical facilities capable of taking insulin measurements.

For now, let's proceed with the imputed data set but we'll remove the problematic `skin` and `serum` variables from consideration.

```
pima_final = pima_impute %>%
    dplyr::select(-serum, -skin) %>%
    dplyr::mutate(y = factor(y))
GGally::ggpairs(pima_final, aes(alpha = 0.05))
```



1.1.1 Logistic regression

2. Fit a logistic regression to the data and perform backward stepwise model selection using the AIC.

```
fm = glm(y ~ ., data = pima_final, family = binomial)
step_model = step(fm, trace = FALSE)
```

```
stargazer::stargazer(fm, step_model, type = "html", column.labels = c("Full model",
    "Stepwise"))
```

	Dependent variable:	
	y	
	Full model	Stepwise
	(1)	(2)
npreg	0.125***	0.143***
	(0.032)	(0.028)
glu	0.036***	0.037***
	(0.004)	(0.003)
bp	-0.010	
	(0.009)	
bmi	0.095***	0.089***
	(0.015)	(0.015)
ped	0.862***	0.883***
	(0.296)	(0.295)
age	0.014	
	(0.009)	
Constant	-9.017***	-9.189***
	(0.806)	(0.706)
Observations	768	768
Log Likelihood	-356.744	-358.084
Akaike Inf. Crit.	727.487	726.168
Note:	*p<0.1;* ***p<0.05;*** p<0.01	

3. In the stepwise model, increases in which variables lead to higher odds of diabetes?

Number of times pregnant, glucose, BMI and diabetes pedigree function are all significantly positively associated with diabetes

4. Write down the fitted stepwise model.

```
library(equatiomatic)
extract_eq(step_model, use_coefs = TRUE, coef_digits = 3)
```

$$\log\left[\frac{\widehat{P(\text{y} = 1)}}{1 - \widehat{P(\text{y} = 1)}}\right] = -9.189 + 0.143(\text{npreg}) + 0.037(\text{glu}) + 0.089(\text{bmi}) + 0.883(\text{ped})$$

5. Predict the log-odds and the probability of a 35 year old woman who has been pregnant twice, with a BMI of 30, blood pressure of 72, glucose of 122 and diabetes pedigree function of 1. Compare it to a 50 year old woman with the same measurements, except a BMI of 40.

We set up a data frame with the "new" data that we want to make predictions for. The new data frame needs to have variable names that match with the variables used in the model we're trying to predict for. It's OK to have extra variables in there, they just won't be used. For example, in the data frame below we've included all the original predictors, but when we run `predict()` on `step_model` only the variables used in `step_model` will be used in the prediction and the others will be ignored (i.e. only `npreg`, `glu`, `bmi`, `ped` will be used).

```
new_data = data.frame(age = c(35, 50),
                      npreg = c(2, 2),
                      bp = c(100, 100),
                      bmi = c(30, 40),
                      glu = c(122, 122),
                      ped = c(1, 1))
```

Our predictions of the log-odds (also known as the logit) are:

```
predict(step_model, new_data, type = "link")
```

```
          1           2
-0.85542760  0.03163625
```

And we can transform these to probability predictions using:

```
predict(step_model, new_data, type = "response")
```

```
        1         2
0.2982955 0.5079084
```

Try calculating these probabilities by hand (substituting the values into the fitted model) rather than using the `predict()` function to make sure you understand what's going on here. Note that there may be some rounding error if you're doing this manually with coefficients that are only reported to 2 or 3 decimal places.

The standard approach to predicting the class outcome is to round these probabilities to zero or one:

```
predict(step_model, new_data, type = "response") %>% round()
```

```
1 2
0 1
```

So using a threshold of 0.5 (rounding down to zero (no-diabetes) if the predicted probability is less than 0.5 and rounding up to one (diabetes) if the predicted probability is greater than 0.5) both individuals are classified as not having diabetes. Note that individual 2 is pretty tricky to classify, their predicted probability is only *just* larger than 0.5.

6. Generate a **confusion matrix** to assess the in-sample accuracy of the predictions from the stepwise model noting that the positive class is the presence of diabetes.

```
library(caret)
preds = factor(round(predict(step_model, type = "response")))
truth = pima_final$y
confusionMatrix(data = preds, reference = truth, positive = "1")
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 443 118
         1  57 150

               Accuracy : 0.7721
                 95% CI : (0.7408, 0.8014)
    No Information Rate : 0.651
    P-Value [Acc > NIR] : 2.172e-13

                  Kappa : 0.4705

 Mcnemar's Test P-Value : 5.745e-06

            Sensitivity : 0.5597
            Specificity : 0.8860
         Pos Pred Value : 0.7246
         Neg Pred Value : 0.7897
             Prevalence : 0.3490
         Detection Rate : 0.1953
   Detection Prevalence : 0.2695
      Balanced Accuracy : 0.7229

       'Positive' Class : 1
```

The overall accuracy is 0.77.

7. What is the sensitivity and specificity of using our stepwise model as a diagnostic tool to predict diabetes?

Looking at the confusion matrix above we might want to reorder the rows and columns so that it matches with our contingency table from lecture 5 where we have the positive class (i.e. testing positive) in the first row and actually having the disease in the first column.

```
res = data.frame(preds, truth)
res %>%
    mutate(preds = factor(preds, levels = c(1, 0)), truth = factor(truth,
        levels = c(1, 0))) %>%
    janitor::tabyl(truth, preds) %>%
    janitor::adorn_title(placement = "top")
```

```
      preds
truth     1   0
    1   150 118
    0    57 443
```

Recall the sensitivity is $P(S^+ \mid D^+)$ i.e. the probability of the test saying you test positive for diabetes, given that you actually have diabetes. From the confusion matrix we can estimate this by looking down the reference column with 1 as the header:

$$\frac{155}{155 + 113} = 0.578$$

Recall the specificity is $P(S^- \mid D^-)$, i.e. the probability of the test saying you test negative for diabetes, given that you actually don't have diabetes. From the confusion matrix, we can estimate this by looking down the reference column with 0 as the header:

$$\frac{439}{439 + 61} = 0.878$$

So our logistic regression diagnostic tool isn't particularly sensitive nor very specific.

8. Perform 5 fold cross validation to get an idea of out of sample accuracy for the stepwise model.

```
set.seed(2018)
caret::train(y ~ npreg + glu + bmi + ped, data = pima_final, method = "glm",
    family = "binomial", trControl = trainControl(method = "cv", number = 5))
```

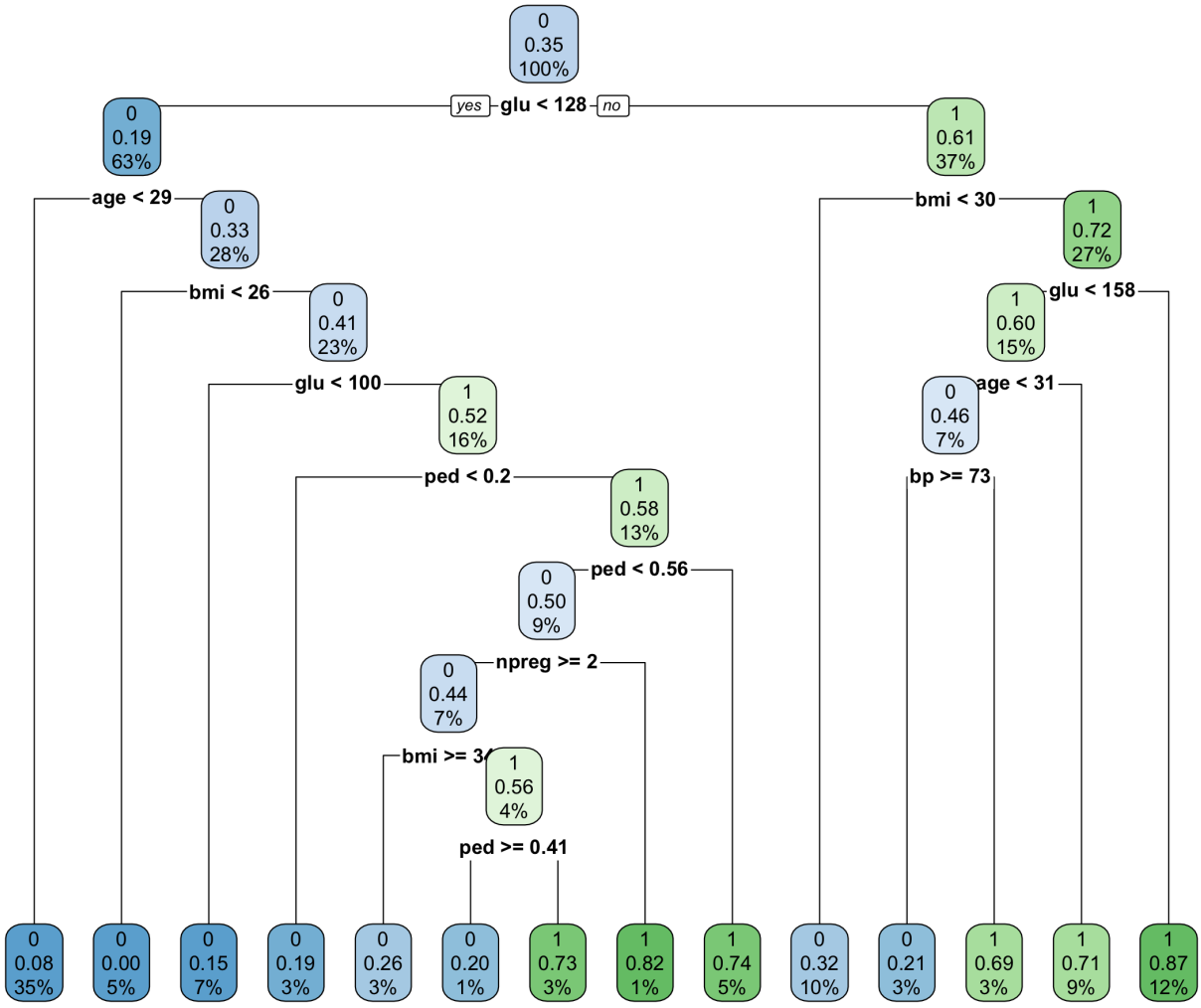
```
Generalized Linear Model

768 samples
  4 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
```

```
Summary of sample sizes: 614, 614, 615, 615, 614
Resampling results:

  Accuracy   Kappa
  0.7669807  0.4589983
```

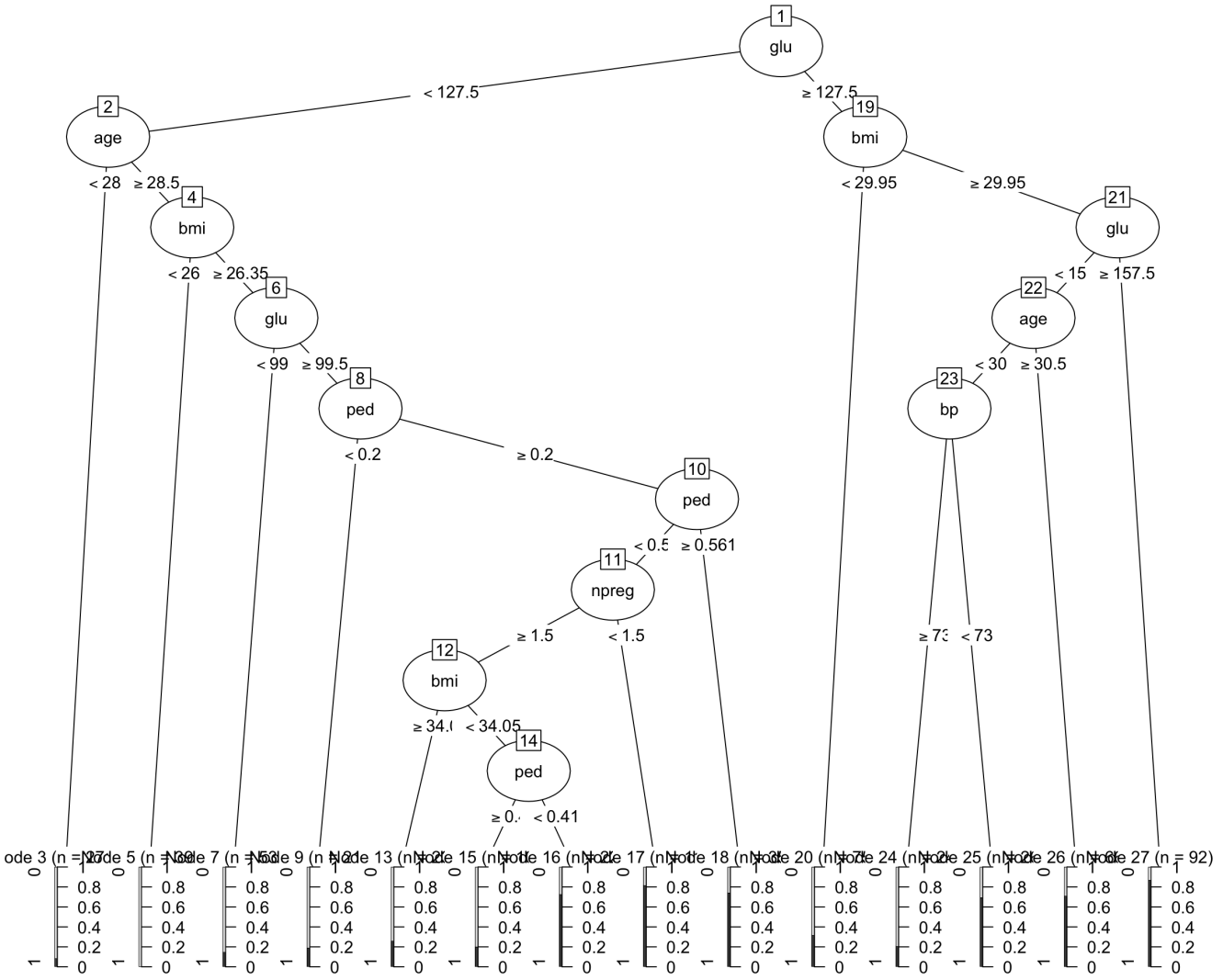
1.1.2 Random forest

9. Fit and visualise a decision tree to this data.

```
library(rpart)
library(rpart.plot)
tree = rpart(y ~ ., data = pima_final)
rpart.plot(tree, cex = 1.1)
```




```
library(partykit)
plot(as.party(tree))
```



10. Predict the outcomes for the two women described earlier using your estimated tree.

```
predict(tree, new_data, type = "class")
```

```
1 2
1 1
Levels: 0 1
```

11. Evaluate in-sample performance using a confusion matrix (e.g. using the `confusionMatrix()` function from the **caret** package).

```
confusionMatrix(predict(tree, type = "class"), truth)
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 444  72
         1  56 196

               Accuracy : 0.8333
                 95% CI : (0.8051, 0.859)
    No Information Rate : 0.651
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.628

 Mcnemar's Test P-Value : 0.1849

            Sensitivity : 0.8880
            Specificity : 0.7313
         Pos Pred Value : 0.8605
         Neg Pred Value : 0.7778
             Prevalence : 0.6510
         Detection Rate : 0.5781
   Detection Prevalence : 0.6719
      Balanced Accuracy : 0.8097

       'Positive' Class : 0
```

The in-sample accuracy is 0.83, a little better than the logistic regression in-sample accuracy of 0.77.

12. Evaluate out-of-sample performance using 5 fold cross validation

```
train(y ~ ., data = pima_final, method = "rpart", trControl = trainControl(method = "cv",
    number = 5))
```

```
CART

768 samples
  6 predictor
  2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 615, 614, 614, 615, 614
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
```

```
  0.01741294  0.7395807  0.4029455
  0.10074627  0.7239963  0.3582183
  0.24253731  0.6849079  0.1810082

Accuracy was used to select the optimal model using the
 largest value.
The final value used for the model was cp = 0.01741294.
```

The tree with the highest out of sample accuracy has a complexity parameter of 0.017. This gave an out of sample accuracy of 0.74, which is a little worse than the logistic regression's 0.77. It appears that the decision tree is over-fitting slightly which drags down its out of sample performance.

13. Fit a random forest and assess out of bag performance.

```
library(randomForest)
set.seed(2018)
rf = randomForest(y ~ ., data = pima_final)
rf
```

```
Call:
 randomForest(formula = y ~ ., data = pima_final)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 23.44%
Confusion matrix:
    0   1 class.error
0 421  79   0.1580000
1 101 167   0.3768657
```

The random forest has an out of bag error rate of 23.44% which corresponds to an out of bag accuracy of 76.56, a little better than the decision tree's accuracy and comparable with the logistic regression model.

1.1.3 Comparison

17. Compare the out of sample accuracies for the different methods using 5 fold CV with 10 repeats.

```
trc = trainControl(method = "repeatedcv",
                   number = 5,
                   repeats = 10)
# decision tree
rpartFit1 = train(y ~ ., data = pima_final,
```

```
                  method = "rpart", trControl = trc)
rpart_acc = max(rpartFit1$results$Accuracy)
# random forests
rfFit1 = train(y ~ ., data = pima_final,
               method = "rf", trControl = trc)
rf_acc = max(rfFit1$results$Accuracy)
# glm
glmFit1 = train(y ~ ., data = pima_final,
                method = "glm", family = "binomial",
                trControl = trc)
glm_acc = glmFit1$results$Accuracy
```

Comparing the results

```
rpart_acc
```

```
[1] 0.74429
```

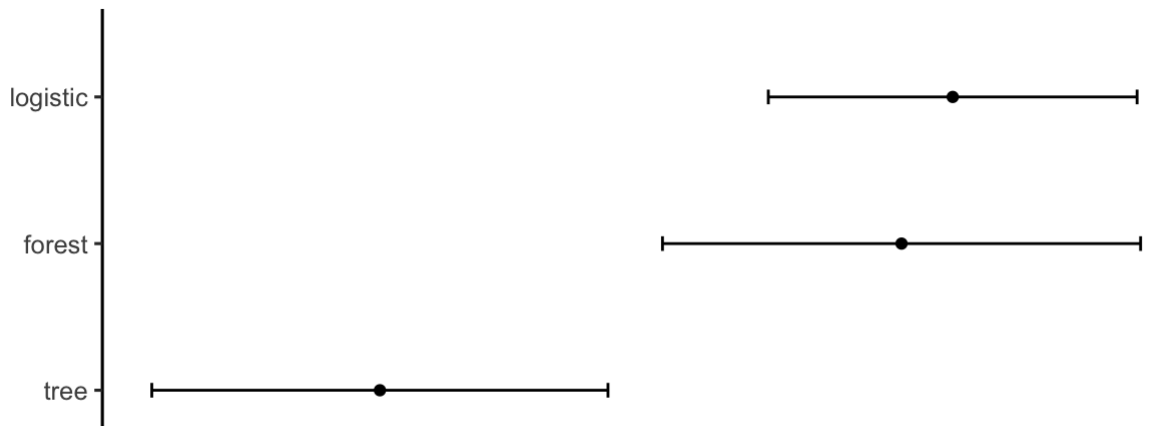
```
rf_acc
```

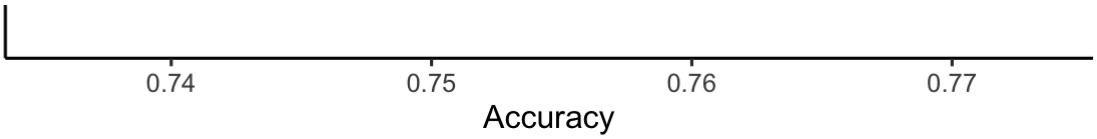
```
[1] 0.7643307
```

```
glm_acc
```

```
[1] 0.7662966
```

```
res = resamples(list(tree = rpartFit1, forest = rfFit1, logistic = glmFit1))
ggplot(res) + labs(y = "Accuracy")
```





A single decision tree performs worst. Logistic regression performed best, closely followed by random forest.

18. Which model do you prefer?

The logistic regression and the random forest performed similarly well. I (Garth) have a preference towards interpretable and transparent models, and so given the similar performance, I would choose the logistic regression.

19. Are our results generalisable to other populations?

Not really. Our data only considered females at least 21 years old of Pima Indian heritage. If we were implementing it as a diagnostic tool, it's only been validated against the similar individuals. In order to extend it more broadly we would need to assess its predictive power on different populations (i.e. people of other heritages).

1.2 Rock wallabies

Macropods defaecate randomly as they forage and scat (faecal pellet). Surveys are a reliable method for detecting the presence of rock-wallabies and other macropods. Scats are used as an indication of spatial foraging patterns of rock-wallabies and sympatric macropods.

Tuft et al. (2011) investigate a rock-wallaby colony in the Warrambungles National Park (NSW). They sampled $n = 200$ sites and recorded the presence or absence of scats as 1 (present) or 0 (absent).

Scats deposited while foraging were not confused with scats deposited while resting because the daytime refuge areas of rock-wallabies were known in detail for each colony and no samples were taken from those areas. Each of the 200 sites were examined separately to account for the different levels of predation risk and the abundance of rock-wallabies.

We will consider five main effects:

- `edible`: Percentage cover of edible vegetation
- `inedible`: Percentage cover of inedible vegetation
- `canopy`: Percentage canopy cover
- `distance`: Distance from diurnal refuge

- `shelter`: Whether or not a plot occurred within a shelter point (large rock or boulder pile)

As well as three interaction terms: - `edible * distance` - `edible * shelter` - `distance * shelter`

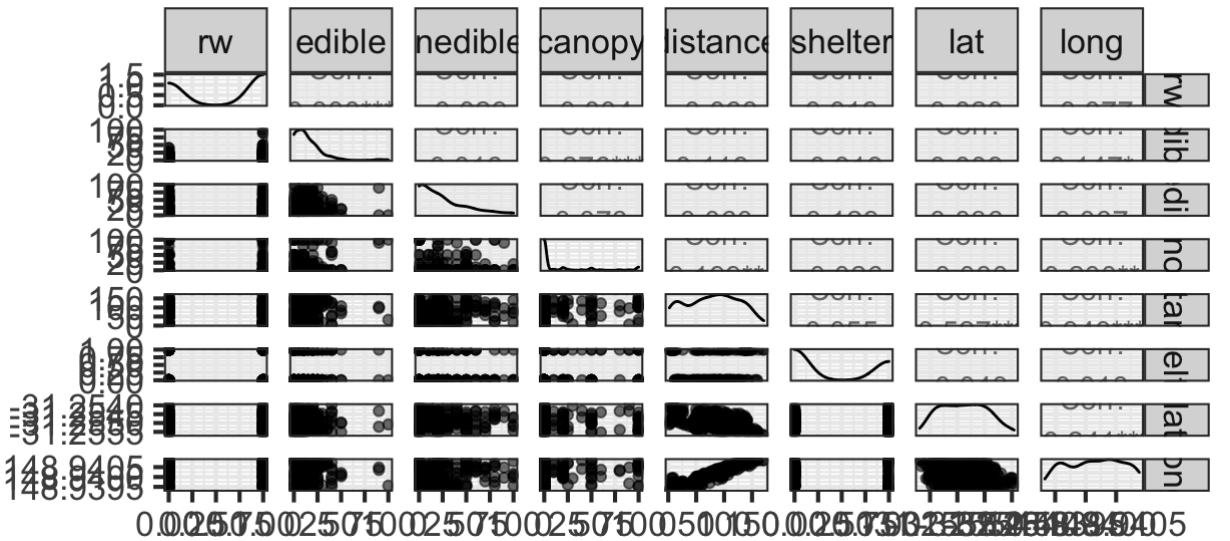
The data can be found in the **mplot** package:

```
# install.packages('mplot')
data("wallabies", package = "mplot")
glimpse(wallabies)
```

```
Rows: 200
Columns: 8
$ rw       <int> 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, …
$ edible   <int> 1, 0, 10, 0, 0, 10, 20, 15, 35, 25, 25, 10, 0, 20, …
$ inedible <int> 15, 0, 0, 50, 10, 50, 15, 50, 25, 30, 60, 100, 25, …
$ canopy   <int> 0, 0, 20, 40, 50, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 20…
$ distance <int> 128, 131, 137, 136, 138, 140, 141, 141, 139, 138, 1…
$ shelter  <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ lat      <dbl> -31.25447, -31.25456, -31.25461, -31.25468, -31.254…
$ long     <dbl> 148.9408, 148.9408, 148.9409, 148.9409, 148.9409, 1…
```

1. Visualise the data.

```
GGally::ggpairs(wallabies, mapping = aes(alpha = 0.2)) + theme_bw(base_size = 16)
```



2. Fit the full logistic regression model including the five main effects and three interaction terms.

```
M1 = glm(rw ~ inedible + canopy + edible * distance + edible * shelter +
    distance * shelter, family = binomial, data = wallabies)
summary(M1)
```

```
Call:
glm(formula = rw ~ inedible + canopy + edible * distance + edible *
    shelter + distance * shelter, family = binomial, data = wallabies)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2762  -1.0810   0.4916   1.0107   1.6596

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       0.1785976  0.5547714   0.322  0.74751
inedible         -0.0035853  0.0060614  -0.591  0.55419
canopy           -0.0017489  0.0056838  -0.308  0.75831
edible            0.1244071  0.0435224   2.858  0.00426 **
distance         -0.0073732  0.0068719  -1.073  0.28329
shelter          -1.1439199  0.7052026  -1.622  0.10478
edible:distance  -0.0006349  0.0004034  -1.574  0.11546
edible:shelter    0.0313602  0.0371986   0.843  0.39920
distance:shelter  0.0118275  0.0076204   1.552  0.12064
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 272.12  on 199  degrees of freedom
Residual deviance: 237.27  on 191  degrees of freedom
AIC: 255.27

Number of Fisher Scoring iterations: 5
```

3. Perform stepwise selection using the AIC from the full model to identify a simpler model.

```
M1_aic = step(M1)
```

```
Start:  AIC=255.27
rw ~ inedible + canopy + edible * distance + edible * shelter +
    distance * shelter

                  Df Deviance    AIC
- canopy           1   237.36 253.36
- inedible         1   237.62 253.62
- edible:shelter   1   238.00 254.00
<none>                 237.27 255.27
- distance:shelter  1   239.71 255.71
- edible:distance   1   239.93 255.93

Step:  AIC=253.36
rw ~ inedible + edible + distance + shelter + edible:distance +
```

```
    edible:shelter + distance:shelter

                   Df Deviance    AIC
- inedible          1   237.73 251.73
- edible:shelter    1   238.10 252.10
<none>                  237.36 253.36
- distance:shelter  1   239.87 253.87
- edible:distance   1   240.09 254.09

Step:  AIC=251.73
rw ~ edible + distance + shelter + edible:distance + edible:shelter +
    distance:shelter

                   Df Deviance    AIC
- edible:shelter    1   238.47 250.47
<none>                  237.73 251.73
- distance:shelter  1   240.07 252.07
- edible:distance   1   240.36 252.36

Step:  AIC=250.47
rw ~ edible + distance + shelter + edible:distance + distance:shelter

                   Df Deviance    AIC
<none>                  238.47 250.47
- edible:distance   1   240.82 250.82
- distance:shelter  1   242.12 252.12
```

```
sjPlot::tab_model(M1, M1_aic, show.ci = FALSE)
```

	rw		rw	
Predictors	*Odds Ratios*	*p*	*Odds Ratios*	*p*
(Intercept)	1.20	0.748	1.10	0.851
inedible	1.00	0.554		
canopy	1.00	0.758		
edible	1.13	**0.004**	1.14	**0.002**
distance	0.99	0.283	0.99	0.136
shelter	0.32	0.105	0.40	0.163
edible * distance	1.00	0.115	1.00	0.136
edible * shelter	1.03	0.399		
distance * shelter	1.01	0.121	1.01	0.060
Observations	200		200	

R^2 Tjur	0.153	0.147

4. Write down the fitted stepwise model.

```
library(equatiomatic)
extract_eq(M1_aic, use_coefs = TRUE, coef_digits = 3)
```

$$\log\left[\frac{\widehat{P(\text{rw} = 1)}}{1 - \widehat{P(\text{rw} = 1)}}\right] = 0.099 + 0.131(\text{edible}) - 0.01(\text{distance}) - 0.928(\text{shelter}) - 0.001(\text{edible} \times$$

5. If you were to use a backward selection p-value approach to drop another variable from the stepwise model selected using AIC, which would you drop?

Shelter has the largest individual p-value (0.163) HOWEVER we wouldn't drop the main effect for shelter when there is an interaction involving shelter still in the model. So if we were to drop a variable it would be the `edible*distance` interaction term as it has the largest p-value (0.136).

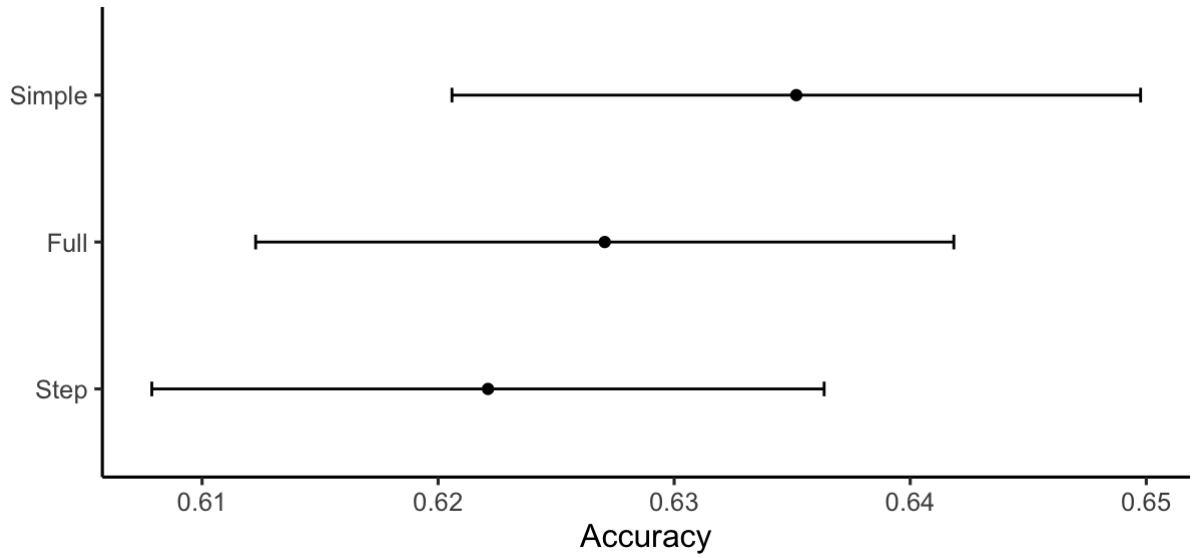
6. Use the **caret** package to compare the out of sample performance of the full model and the stepwise model with a simple model that only uses `edible` as a predictor. Use repeated 10-fold cross validation with 20 repeats.

```
library(caret)
set.seed(2021)
trc = trainControl(method = "repeatedcv", number = 10, repeats = 20)
M1_caret = caret::train(factor(rw) ~ inedible + canopy + edible * distance +
    edible * shelter + distance * shelter, data = wallabies, method = "glm",
    family = "binomial", trControl = trc)
M1_step_caret = caret::train(factor(rw) ~ edible + distance + shelter +
    edible:distance + distance:shelter, data = wallabies, method = "glm",
    family = "binomial", trControl = trc)
M1_simple = caret::train(factor(rw) ~ edible, data = wallabies, method = "glm",
    family = "binomial", trControl = trc)
res = resamples(list(Full = M1_caret, Step = M1_step_caret, Simple = M1_simple))
summary(res)
```

```
Call:
summary.resamples(object = res)

Models: Full, Step, Simple
Number of resamples: 200

Accuracy
           Min.   1st Qu.    Median      Mean 3rd Qu.      Max. NA's
```

```
Full   0.4210526 0.5500000 0.6190476 0.6270583     0.7 0.9473684    0
Step   0.3500000 0.5500000 0.6315789 0.6221122     0.7 0.9047619    0
Simple 0.3000000 0.5714286 0.6315789 0.6351729     0.7 0.9047619    0

Kappa
             Min.    1st Qu.    Median      Mean   3rd Qu.      Max.
Full   -0.2514970 0.08163265 0.2222222 0.2282786 0.3739698 0.8901734
Step   -0.3265306 0.07951717 0.2264957 0.2179259 0.3478261 0.8000000
Simple -0.3861386 0.08992095 0.2312139 0.2347194 0.3852496 0.8000000
       NA's
Full      0
Step      0
Simple    0
```

```
ggplot(res, metric = "Accuracy") + labs(y = "Accuracy")
```



The simple model actually has the highest (out of sample) accuracy, though it should be noted that none of the models are very accurate. Note that the baseline accuracy rate is 58% and the accuracies from these models are not much higher than that.

```
wallabies %>%
    janitor::tabyl(rw)
```

```
 rw   n percent
  0  84    0.42
  1 116    0.58
```

2 Additional resources

For more details on decision trees see Hastie, Tibshirani, and Friedman (2009, sec. 9.2) and James et al. (2017, chap. 8).

As additional practice questions, I recommend these two DataCamp chapters:

- </> Logistic Regression
- </> Decision trees

References

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. 2nd ed. Springer Series in Statistics. New York, NY, USA: Springer. https://web.stanford.edu/~hastie/ElemStatLearn/.

Hlavac, Marek. 2018. *Stargazer: Well-Formatted Regression and Summary Statistics Tables*. Bratislava, Slovakia: Central European Labour Studies Institute (CELSI). https://CRAN.R-project.org/package=stargazer.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2017. *An Introduction to Statistical Learning: With Applications in r*. New York: Springer. https://www-bcf.usc.edu/~gareth/ISL/.

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2018. *Caret: Classification and Regression Training*. https://CRAN.R-project.org/package=caret.

Johannes, R S. 1988. "Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus." *Johns Hopkins APL Technical Digest* 10: 262–66.

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by randomForest." *R News* 2 (3): 18–22. https://CRAN.R-project.org/doc/Rnews/.

Therneau, Terry, and Beth Atkinson. 2018. *Rpart: Recursive Partitioning and Regression Trees*. https://CRAN.R-project.org/package=rpart.

Tuft, K. D., M. S. Crowther, K. Connell, S. Müller, and C. McArthur. 2011. "Predation Risk and Competitive Interactions Affect Foraging of an Endangered Refuge-Dependent Herbivore." *Animal Conservation* 14 (4): 447–57. https://doi.org/10.1111/j.1469-1795.2011.00446.x.