

Logistic Regression

Mehran Karimzadeh

September 11, 2018

Data and packages for these slides:

```
knitr::opts_chunk$set(echo = FALSE)
# required_packages = c("caret", "tree", "randomForest",
#                        "cowplot", "e1071")
# install.packages(required_packages)
suppressMessages(require(tidyverse))
suppressMessages(require(cowplot))
mice_df = read_csv("mice.csv")
```

```
## Parsed with column specification:
## cols(
##   Age = col_double(),
##   Sex = col_character(),
##   Condition = col_character(),
##   Mouse.Genotyping = col_character(),
##   ID = col_integer(),
##   Timepoint = col_character(),
##   Genotype = col_character(),
##   DaysOfEE = col_integer(),
##   DaysOfEE0 = col_integer()
## )
```

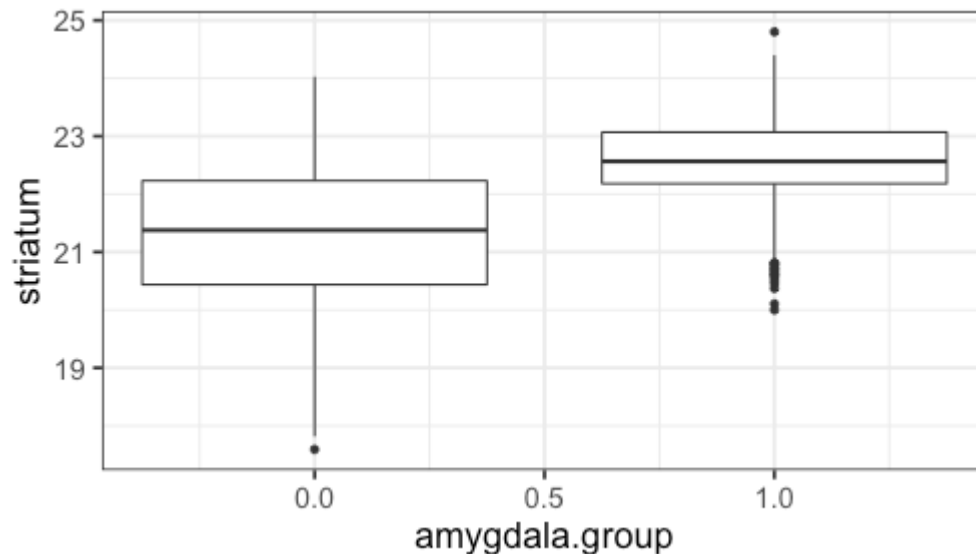
```
volume_df = read_csv("volumes.csv")
```

Logistic regression

Binary variables

Can we predict gender given striatum volume?

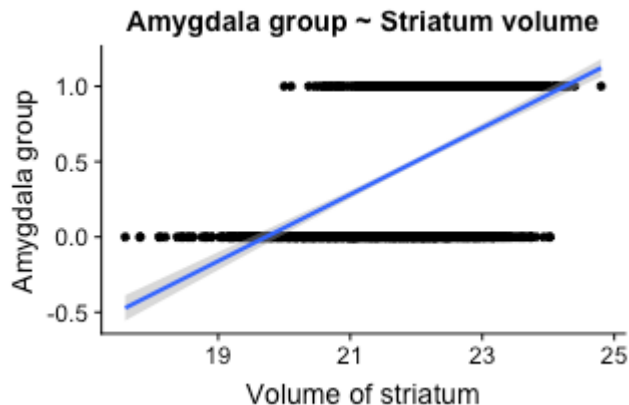
```
mice$amygdala.group = ifelse(mice$amygdala > 10, 1, 0)
ggplot(mice, aes(x=amygdala.group, y=striatum,
                 group=amygdala.group)) +
  geom_boxplot() +
  theme_bw(base_size=18)
```



Linear model for binary variables?

- If the independent variable is binary, can we fit the linear model?

```
ggplot(mice, aes(x=striatum, y=amygdala.group)) +  
  geom_point() + xlab("Volume of striatum") +  
  ylab("Amygdala group") +  
  geom_smooth(method="lm") +  
  ggtitle("Amygdala group ~ Striatum volume")
```



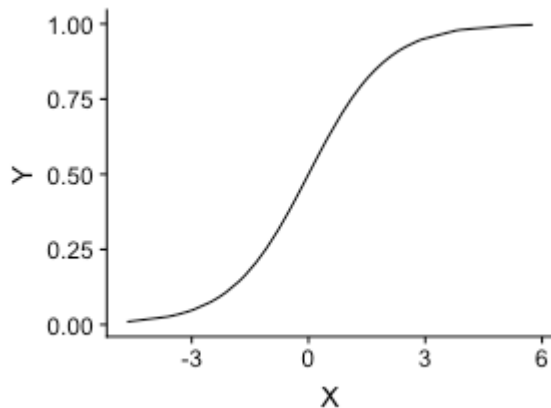
Why we can't use linear model for classification?

- Suppose we want to predict seizure, stroke, or overdose given some measurements from patients
- If we model them as 1, 2, and 3 respectively, we are assuming order
- Even in case of binary variables, our estimates may exceed range of $[0, 1]$, making the interpretation unnecessarily hard
- Any other reasons that contradict assumptions of the linear model?

Logistic function

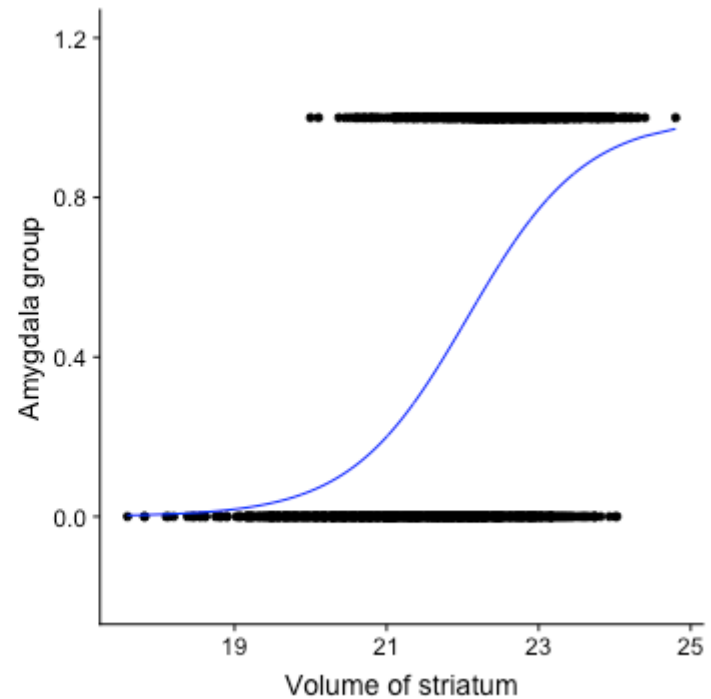
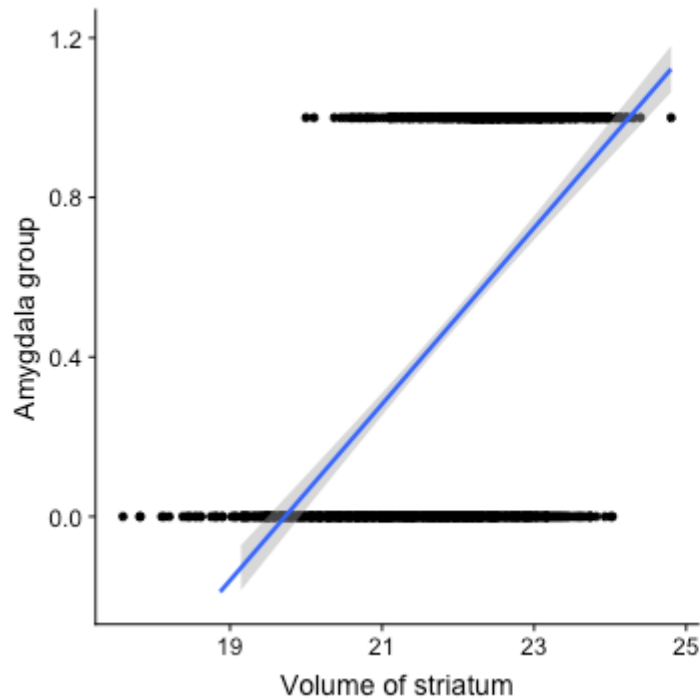
- $\frac{L}{1+e^{-k*(x-\sigma_0)}}$

```
logistic_function = function(input, curve_max=1,  
                              curve_steepness=1, sig_mid=0){  
  output = curve_max /  
    (1 + exp(-curve_steepness * (input - sig_mid)))  
}  
input_vals = rnorm(50, sd=3)  
out_df = data.frame(  
  X=input_vals, Y=logistic_function(input_vals))  
ggplot(out_df, aes(x=X, y=Y)) + geom_line()
```



Linear model for binary variables?

Warning: Removed 14 rows containing missing values (geom_smooth).



Solving the logistic model

- $p(X) = \beta_0 + \beta X$

Solving the logistic model

- $p(X) = \beta_0 + \beta X$
- $p(X) = \frac{1}{1+e^{\beta_0+\beta X}} \rightarrow$ estimating probability with logistic function

Solving the logistic model

- $p(X) = \beta_0 + \beta X$
- $p(X) = \frac{1}{1+e^{\beta_0+\beta X}} \rightarrow$ estimating probability with logistic function
- $\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta X} \rightarrow$ odds

Solving the logistic model

- $p(X) = \beta_0 + \beta X$
- $p(X) = \frac{1}{1+e^{\beta_0+\beta X}} \rightarrow$ estimating probability with logistic function
- $\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta X} \rightarrow$ odds
- $\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta X \rightarrow$ logit or log of odds

Solving the logistic model

- $p(X) = \beta_0 + \beta X$
- $p(X) = \frac{1}{1+e^{\beta_0+\beta X}} \rightarrow$ estimating probability with logistic function
- $\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta X} \rightarrow$ odds
- $\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta X \rightarrow$ logit or log of odds
- In linear model, β shows how a unit increase in X changes Y

Solving the logistic model

- $p(X) = \beta_0 + \beta X$
- $p(X) = \frac{1}{1+e^{\beta_0+\beta X}} \rightarrow$ estimating probability with logistic function
- $\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta X} \rightarrow$ odds
- $\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta X \rightarrow$ logit or log of odds
- In linear model, β shows how a unit increase in X changes Y
- The effect size β shows how a unit increase in X changes log odds

Solving the logistic model

- $p(X) = \beta_0 + \beta X$
- $p(X) = \frac{1}{1+e^{\beta_0+\beta X}} \rightarrow$ estimating probability with logistic function
- $\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta X} \rightarrow$ odds
- $\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta X \rightarrow$ logit or log of odds
- In linear model, β shows how a unit increase in X changes Y
- The effect size β shows how a unit increase in X changes log odds
- In linear regression, we used least squared to minimize mean squared error

Solving the logistic model

- $p(X) = \beta_0 + \beta X$
- $p(X) = \frac{1}{1+e^{\beta_0+\beta X}} \rightarrow$ estimating probability with logistic function
- $\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta X} \rightarrow$ odds
- $\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta X \rightarrow$ logit or log of odds
- In linear model, β shows how a unit increase in X changes Y
- The effect size β shows how a unit increase in X changes log odds
- In linear regression, we used least squared to minimize mean squared error
- In logistic regression, we use **maximum likelihood**

Solving the logistic model

- $p(X) = \beta_0 + \beta X$
- $p(X) = \frac{1}{1+e^{\beta_0+\beta X}} \rightarrow$ estimating probability with logistic function
- $\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta X} \rightarrow$ odds
- $\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta X \rightarrow$ logit or log of odds
- In linear model, β shows how a unit increase in X changes Y
- The effect size β shows how a unit increase in X changes log odds
- In linear regression, we used least squared to minimize mean squared error
- In logistic regression, we use **maximum likelihood**
- $l(\beta_0, \beta) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} (1 - p(x_{i'})) \rightarrow$ Likelihood function

K-nearest neighbours

- Example of a non-parametric, simple, and powerful machine learning method

K-NN

- Given a positive integer K and a datapoint x_0 , identifies K points in training data which are closest to a datapoint x_0 .

K-NN

- Given a positive integer K and a datapoint x_0 , identifies K points in training data which are closest to a datapoint x_0 .
- It then estimates the conditional probability for label of x_0 given responses for its K nearest neighbours

K-NN

- Given a positive integer K and a datapoint x_0 , identifies K points in training data which are closest to a datapoint x_0 .
- It then estimates the conditional probability for label of x_0 given responses for its K nearest neighbours

*Let's implement it in R!

K-NN algorithm

- Split data to training and test

K-NN algorithm

- Split data to training and test

```
split_ratio = 0.8
idx_train = sample(1:nrow(mice),
                   size=floor(nrow(mice) * split_ratio))
train_df = mice[idx_train, ]
test_df = mice[-idx_train, ]
```

- Predict amygdala size given volume of striatum and midbrain

- Predict amygdala size given volume of striatum and midbrain
- Finding nearest neighbours

```
get_neighbours = function(test_data, train_df, K=5){  
  print(test_data)  
  merged_df = rbind(test_data, train_df)  
  dist_df = as.matrix(dist(merged_df))  
  distances = as.numeric(dist_df[1, ])  
  idx_out = order(distances, decreasing=FALSE)[2:(K + 1)]  
  return(idx_out)  
}
```

K-NN prediction

```
predictive_features = c("striatum", "midbrain")
response = "amygdala.group"
test_df$Posterior = NA
for(i in 1:nrow(test_df)){
  idx_neighbours = get_neighbours(
    test_df[i, predictive_features],
    train_df[, predictive_features])
  labels = unlist(train_df[idx_neighbours, response])
  prob = mean(labels)
  test_df$Posterior[i] = prob
}
```

```
## # A tibble: 1 x 2
##   striatum midbrain
##   <dbl>     <dbl>
## 1    22.0     13.8
## # A tibble: 1 x 2
##   striatum midbrain
##   <dbl>     <dbl>
## 1    23.0     14.0
## # A tibble: 1 x 2
##   striatum midbrain
```

Calculating threshold-based metrics

```
suppressMessages(require(caret))
suppressMessages(require(e1071))
confMat = confusionMatrix(
  factor(test_df$Posterior > 0.5), factor(test_df$amygdala.group == 1)
)
print(as.data.frame(confMat$byClass))
```

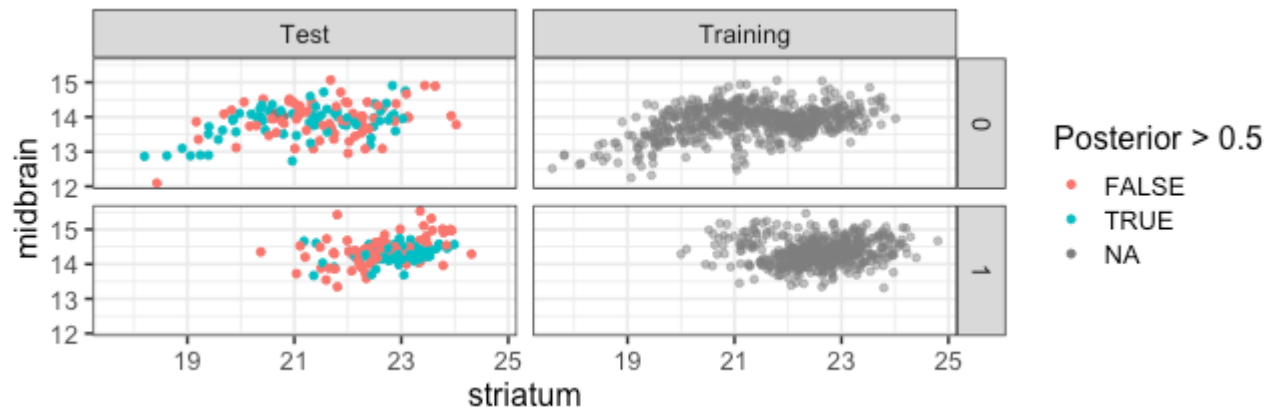
```
##                                confMat$byClass
## Sensitivity                    0.5072464
## Specificity                    0.4637681
## Pos Pred Value                 0.4861111
## Neg Pred Value                 0.4848485
## Precision                      0.4861111
## Recall                        0.5072464
## F1                           0.4964539
## Prevalence                    0.5000000
## Detection Rate                 0.2536232
## Detection Prevalence          0.5217391
## Balanced Accuracy              0.4855072
```

```
print(paste("Accuracy =", signif(confMat$overall["Accuracy"], 3)))
```

```
## [1] "Accuracy = 0.486"
```

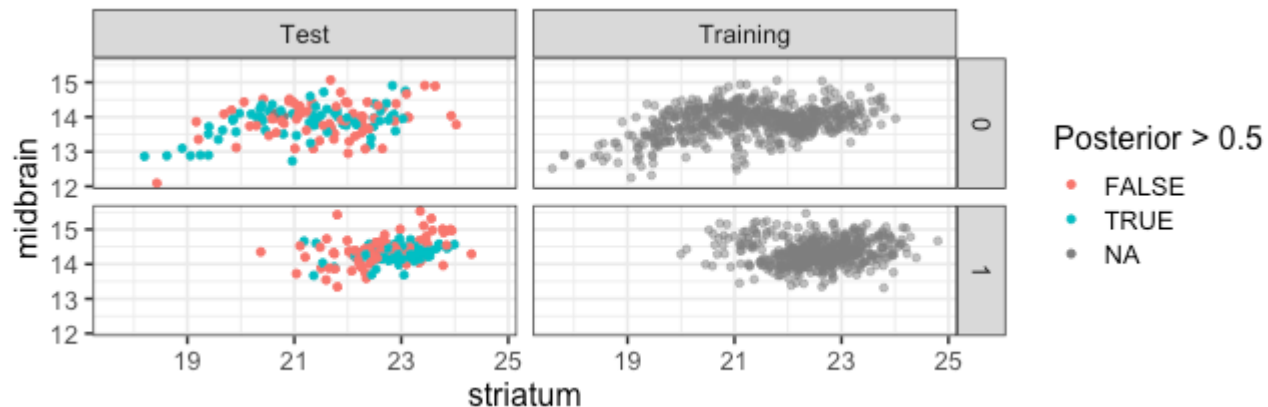
Plotting performance

```
train_df$Posterior = NA
train_df$Dataset = "Training"
test_df$Dataset = "Test"
merged_df = rbind(train_df, test_df)
ggplot(merged_df) +
  aes(x=striatum, y=midbrain, colour=Posterior > 0.5) +
  geom_point(alpha=0.5) +
  geom_point(data=test_df, aes(colour=Posterior > 0.5)) +
  theme_bw(base_size=16) +
  facet_grid(factor(amygdala.group)~Dataset)
```



Plotting performance

```
train_df$Posterior = NA
train_df$Dataset = "Training"
test_df$Dataset = "Test"
merged_df = rbind(train_df, test_df)
ggplot(merged_df) +
  aes(x=striatum, y=midbrain, colour=Posterior > 0.5) +
  geom_point(alpha=0.5) +
  geom_point(data=test_df, aes(colour=Posterior > 0.5)) +
  theme_bw(base_size=16) +
  facet_grid(factor(amygdala.group)~Dataset)
```



Random forest, in simple terms

- Random forests can regress or classify, and they are made of hundreds of trees

Random forest, in simple terms

- Random forests can regress or classify, and they are made of hundreds of trees
- Each tree uses some of data (samples) and some of the features

Random forest, in simple terms

- Random forests can regress or classify, and they are made of hundreds of trees
- Each tree uses some of data (samples) and some of the features
- We identify which feature can classify (or regress) the outcome better

Random forest, in simple terms

- Random forests can regress or classify, and they are made of hundreds of trees
- Each tree uses some of data (samples) and some of the features
- We identify which feature can classify (or regress) the outcome better
- We split the data at the point which classifies training data best, and repeat the last step on each split until all data points are grouped

Random forest, in simple terms

- Random forests can regress or classify, and they are made of hundreds of trees
- Each tree uses some of data (samples) and some of the features
- We identify which feature can classify (or regress) the outcome better
- We split the data at the point which classifies training data best, and repeat the last step on each split until all data points are grouped
- We build hundreds of trees based on training data. When it comes to new data, we use the majority vote to decide on the response for output variable

Classification tree

```
require(tree)
```

```
## Loading required package: tree
```

```
volume_df = read.csv("volumes.csv")
volume_df = volume_df[, !colnames(volume_df) %in% c("ID", "Timepoint")]
volume_df$amygdala.group = ifelse(volume_df$amygdala > 10, 1, 0)
train_df = volume_df[idx_train, ]
test_df = volume_df[-idx_train, ]
tree_model = tree(factor(amygdala.group) ~.-amygdala, train_df)
summary(tree_model)
```

```
##
```

```
## Classification tree:
```

```
## tree(formula = factor(amygdala.group) ~ . - amygdala, data = train_df)
```

```
## Variables actually used in tree construction:
```

```
## [1] "hippocampus"
```

```
## [2] "Posteromedial.cortical.amygdaloid.area"
```

```
## [3] "midbrain"
```

```
## [4] "hypothalamus"
```

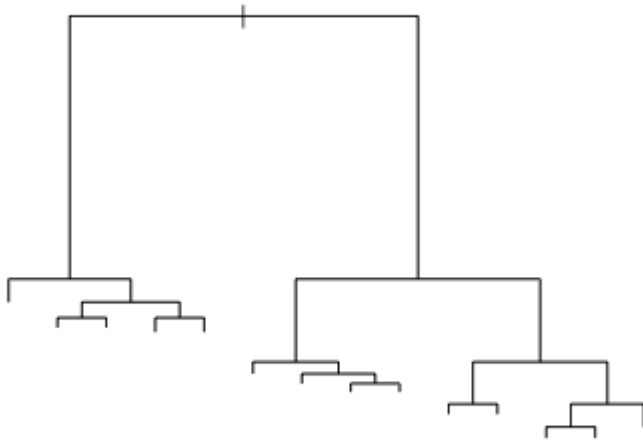
```
## [5] "Ventral.nucleus.of.the.endopiriform.clastrum"
```

```
## [6] "Secondary.visual.cortex..mediolateral.area"
```

```
pred_tree = predict(tree_model, test_df, type="class")
confMat = confusionMatrix(pred_tree, factor(test_df$amygdala.group))
acc_tree = signif(confMat$overall["Accuracy"], 3)
print(paste("Accuracy =", acc_tree))
```

```
## [1] "Accuracy = 0.846"
```

```
plot(tree_model)
```



Fitting a random forest

```
suppressMessages(require(randomForest))
rf_model = randomForest(factor(amygdala.group) ~.-amygdala,
                        data=train_df, ntree=100, importance=TRUE)
pred_rf = predict(rf_model, newdata=test_df)
confMat = confusionMatrix(pred_rf, factor(test_df$amygdala.group))
acc_rf = signif(confMat$overall["Accuracy"], 3)
print(paste("Accuracy RF =", acc_rf, "and tree =", acc_tree))
```

```
## [1] "Accuracy RF = 0.903 and tree = 0.846"
```

Feature importance by random forest

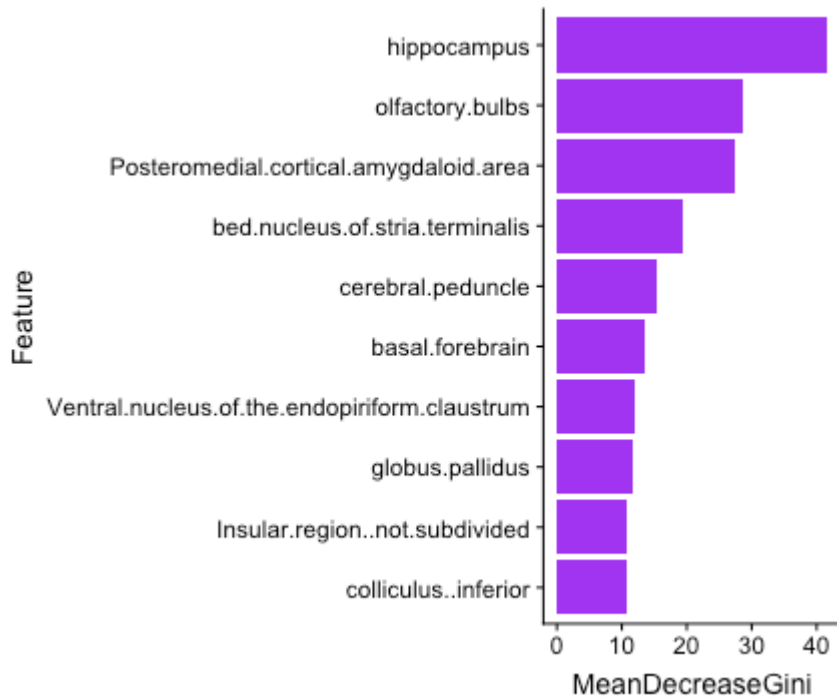
- Mean decrease Gini is the sum of Gini impurity of a feature across all trees.

Feature importance by random forest

- Mean decrease Gini is the sum of Gini impurity of a feature across all trees.
- Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled

```
imp_df = as.data.frame(importance(rf_model))
imp_df$Feature = rownames(imp_df)
imp_df = imp_df[order(imp_df$MeanDecreaseGini, decreasing=TRUE)[1:10]]
imp_df$Feature = factor(
  imp_df$Feature,
  levels=imp_df$Feature[order(imp_df$MeanDecreaseGini)])
```

```
ggplot(imp_df) +  
  aes(x=Feature, y=MeanDecreaseGini) +  
  geom_bar(stat="identity", fill="purple") +  
  coord_flip()
```



Assignment

1. Compare accuracy of K-NN with logistic regression in predicting amygdala size on test dataset.
2. Split mice data into training, test, and validation. Optimize K hyperparameter of K-NN so the model has a higher performance in predicting amygdala size on test dataset. Report at least two different measures of binary classification on validation data. Use knn function in R.
3. Use random forest to predict genotype given volume of all regions in the brain. Optimize the number of trees in the random forest. Compare your training and validation accuracy, specificity, and sensitivity for each genotype (with a plot). Which variables have the highest feature importance (show in a plot)?
4. Why do we split data into training, test, and validation?