Model testing

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(latex2exp)

library(class)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(rpart)
library(rpart.plot)

library(ROCR)

set.seed(1101)

.save_and_display <- function(g, main, file, width = 960, height = 480, ...) {
    g <- g +
        ggtitle(main) +
        theme(
            text = element_text(size = 12),
            plot.title = element_text(hjust = 0.5, size = 16),
            strip.text = element_text(size = 12),
            # legend.position="bottom"
        )

    ggsave(file, plot = g, units = 'px', width = width, height = height, dpi = 100, ...)
    g
}
```

Input dataframe

```
df <- read.csv('heart-disease-dsa1101.csv')

CATEGORICAL_VARIABLES <- c(
    'sex',
    'chest.pain',
    'fbs',
    'rest.ecg',
    'angina',
    'blood.disorder'
)

RESPONSE <- 'disease'

NUMERICAL_VARAIBLES <- c(
    'age',
    'bp',
    'chol',
    'heart.rate',
    'st.depression',
    'vessels'
    # 'vessels' is a discrete small variable from raning from 0-4
)

df <- df %>%
    filter(blood.disorder != 0) %>%
    mutate_at(all_of(c(CATEGORICAL_VARIABLES, RESPONSE)), as.factor)
```

```
## Warning: Using `all_of()` outside of a selecting function was deprecated in tidyselect
## 1.2.0.
## i See details at
##   <https://tidyselect.r-lib.org/reference/faq-selection-context.html>
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
str(df)
```

```
## 'data.frame':    298 obs. of  13 variables:
##  $ age           : int  63 37 41 56 57 57 56 44 52 57 ...
##  $ sex           : Factor w/ 2 levels "0","1": 2 2 1 2 1 2 1 2 2 2 ...
##  $ chest.pain    : Factor w/ 4 levels "0","1","2","3": 4 3 2 2 1 1 2 2 3 3 ...
##  $ bp            : int  145 130 130 120 120 140 140 120 172 150 ...
##  $ chol          : int  233 250 204 236 354 192 294 263 199 168 ...
##  $ fbs           : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 2 1 ...
##  $ rest.ecg      : Factor w/ 3 levels "0","1","2": 1 2 1 2 2 2 1 2 2 2 ...
##  $ heart.rate    : int  150 187 172 178 163 148 153 173 162 174 ...
##  $ angina        : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
##  $ st.depression : num  2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
##  $ vessels       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ blood.disorder: Factor w/ 3 levels "1","2","3": 1 2 2 2 2 1 2 3 3 2 ...
##  $ disease       : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Dummy categorical variables

```
.dummies <- function(df, col) {
    df_col <- df %>%
        select(all_of(col)) %>% pull()

    lvls <- levels(df_col)
    dummy_cols <- paste(col, lvls, sep='.')

    dummies <- lvls %>% sapply(function(lvl) { as.numeric(df_col == lvl) }) %>%
        data.frame()

    colnames(dummies) <- dummy_cols
    dummies
}

.mutate_dummies <- function(df, cols) {
    for (col in cols)
        df <- cbind(df, .dummies(df, col))
    select(df, -all_of(cols))
}
```

Data normalisation / standardisation

```
.minmaxscale <- function(xs) {
    min.xs <- min(xs)
    max.xs <- max(xs)
    (xs - min.xs) / (max.xs - min.xs)
}

.boxcox <- function(xs, LAMBDA = 0) {
    sapply(xs, function(x) {
        if (LAMBDA == 0)
            return (log(x))
        return (x ** LAMBDA - 1) / LAMBDA
    })
}

.mutate_boxcox <- function(df) {
    df %>%
        mutate(age        = .boxcox(age,         1.505)) %>%
        mutate(bp         = .boxcox(bp,         -0.645)) %>%
        mutate(chol       = .boxcox(chol,       -0.125)) %>%
        mutate(heart.rate = .boxcox(heart.rate,  2.345))
}
```

5-fold cross validation set-up

```
.n_folds = 5
.folds = sample(1:.n_folds, nrow(df), replace=TRUE)
```

```
.predict.fold <- function(fold, pipe, df) {
    ids <- which(.folds == fold)
    train <- df[-ids,]
    test  <- df[ ids,]
    tmp <- pipe(train, test)
    y_pred <- tmp$result
    y_test <- select(test, all_of(RESPONSE)) %>% pull() %>%
        as.character() %>% as.numeric()
    list(y_pred = y_pred, y_test = y_test)
}

.cv <- function(pipe, df) {
    evals <- 1:.n_folds %>%
        as.list() %>%
        lapply(.predict.fold, pipe, df) %>%
        transpose()
    prediction(evals$y_pred, evals$y_test)
}

.all <- function(pipe, df) {
    y_pred <- pipe(df, df)$result
    y_test <- select(df, all_of(RESPONSE)) %>% pull() %>%
        as.character() %>% as.numeric()
    prediction(y_pred, y_test)
}

.predict.all    <- \(params, model, df) params %>% lapply(model) %>% lapply(.all, df)
.predict.all.cv <- \(params, model, df) params %>% lapply(model) %>% lapply(.cv,  df)

.single.metric <- function(preds, measure) {
    preds %>%
        lapply(performance, measure = measure) %>%
        lapply(\(pref)
            pref@y.values %>%
            lapply(median) %>% unlist() %>% mean()
        ) %>% unlist()
}
```

k-NN

```
.mutate_all_numeric <- function(df) {
    df %>%
        # mutate(blood.disorder = factor(blood.disorder, levels=c(1, 2, 3), labels = c(1, 3, 2))) %>%
        mutate(across(everything(), \(x) as.double(as.character(x))))
}

NOMINAL_VARIABLES <- c(
    # 'rest.ecg',
    # 'blood.disorder'
    # 'chest.pain'
)
```

```r
df_knn <- df %>%
    # .mutate_dummies(NOMINAL_VARIABLES) %>%
    .mutate_all_numeric()

df_knn_scaled_boxcox <- df %>%
    # .mutate_dummies(NOMINAL_VARIABLES) %>%
    .mutate_all_numeric() %>%
    .mutate_boxcox() %>%
    mutate(across(-all_of(RESPONSE), scale))

df_knn_scaled <- df %>%
    # .mutate_dummies(NOMINAL_VARIABLES) %>%
    .mutate_all_numeric() %>%
    mutate(across(-all_of(RESPONSE), scale))

df_knn_normalised <- df %>%
    # .mutate_dummies(NOMINAL_VARIABLES) %>%
    .mutate_all_numeric() %>%
    mutate(across(-all_of(RESPONSE), .minmaxscale))
```

```r
knn.pipe <- function(k, prob=FALSE, excludes=character(0)) {
    function(train, test) {
        X_train <- train %>% select(-all_of(RESPONSE), -all_of(excludes))
        X_test  <- test  %>% select(-all_of(RESPONSE), -all_of(excludes))
        y_train <- train %>% select( all_of(RESPONSE)) |> pull()
        y_pred <- knn(X_train, X_test, y_train, k, prob=TRUE)

        y <- y_pred %>% as.character() %>% as.numeric()
        if (prob) {
            p <- attr(y_pred, 'prob')
            result <- y * p + (1 - y) * (1 - p)
        } else { result <- y }

        list(result = result, model = k)
    }
}

knn.model            <- \(k, prob=FALSE) knn.pipe(k, prob)
knn.model.simplified <- \(k, prob=FALSE) knn.pipe(k, prob, excludes=c('fbs', 'bp', 'chol'))
knn.ks <- seq(3, 50, 1)

knn.preds.scaled        <- .predict.all.cv(knn.ks, knn.model, df_knn_scaled)
knn.preds.normalised    <- .predict.all.cv(knn.ks, knn.model, df_knn_normalised)
knn.preds.scaled.boxcox <- .predict.all.cv(knn.ks, knn.model, df_knn_scaled_boxcox)

knn.preds.scaled.simplified        <- .predict.all.cv(knn.ks, knn.model.simplified, df_knn_scaled)
knn.preds.normalised.simplified    <- .predict.all.cv(knn.ks, knn.model.simplified, df_knn_normalised)
knn.preds.scaled.boxcox.simplified <- .predict.all.cv(knn.ks, knn.model.simplified, df_knn_scaled_boxco

knn.tprs.scaled        <- knn.preds.scaled        %>% .single.metric('tpr')
knn.tprs.normalised    <- knn.preds.normalised    %>% .single.metric('tpr')
knn.tprs.scaled.boxcox <- knn.preds.scaled.boxcox %>% .single.metric('tpr')
knn.tprs.scaled.simplified        <- knn.preds.scaled.simplified        %>% .single.metric('tpr')
knn.tprs.normalised.simplified    <- knn.preds.normalised.simplified    %>% .single.metric('tpr')
```

```r
knn.tprs.scaled.boxcox.simplified <- knn.preds.scaled.boxcox.simplified %>% .single.metric('tpr')

knn.metrics <- data.frame(
    k = knn.ks,
     # TPR.scaled            =        knn.tprs.scaled,
    # CTPR.scaled            = cummax(knn.tprs.scaled),
     TPR.normalised        =        knn.tprs.normalised,
    CTPR.normalised        = cummax(knn.tprs.normalised),
     TPR.scaled_boxcox     =        knn.tprs.scaled.boxcox,
    CTPR.scaled_boxcox     = cummax(knn.tprs.scaled.boxcox),
    #  TPR.scaled_sim        =        knn.tprs.scaled.simplified,
    # CTPR.scaled_sim        = cummax(knn.tprs.scaled.simplified),
     TPR.normalised_sim    =        knn.tprs.normalised.simplified,
    CTPR.normalised_sim    = cummax(knn.tprs.normalised.simplified),
     TPR.scaled_boxcox_sim =        knn.tprs.scaled.boxcox.simplified,
    CTPR.scaled_boxcox_sim = cummax(knn.tprs.scaled.boxcox.simplified)
) %>%
    pivot_longer(
        cols = starts_with(c('TPR', 'CTPR')),
        names_to = 'metric',
        values_to = 'value'
    ) %>%
    mutate(transformation = gsub('.*\\.', '', metric)) %>%
    mutate(metric = gsub('\\..*', '', metric))

knn.metrics
```

```
## # A tibble: 384 x 4
##        k metric value transformation
##    <dbl> <chr>  <dbl> <chr>
##  1     3 TPR    0.855 normalised
##  2     3 TPR    0.849 scaled_boxcox
##  3     3 TPR    0.863 normalised_sim
##  4     3 TPR    0.854 scaled_boxcox_sim
##  5     3 CTPR   0.855 normalised
##  6     3 CTPR   0.849 scaled_boxcox
##  7     3 CTPR   0.863 normalised_sim
##  8     3 CTPR   0.854 scaled_boxcox_sim
##  9     4 TPR    0.849 normalised
## 10     4 TPR    0.867 scaled_boxcox
## # i 374 more rows
```

```r
BEST.K <- 19
BEST.MODEL <- knn.metrics %>% filter(
    # transformation == 'scaled_boxcox_sim',
    metric == 'CTPR', k == BEST.K
)
BEST.MODEL
```

```
## # A tibble: 4 x 4
##       k metric value transformation
##   <dbl> <chr>  <dbl> <chr>
## 1    19 CTPR   0.863 normalised
## 2    19 CTPR   0.902 scaled_boxcox
## 3    19 CTPR   0.887 normalised_sim
```

```
## 4    19 CTPR   0.932 scaled_boxcox_sim
```

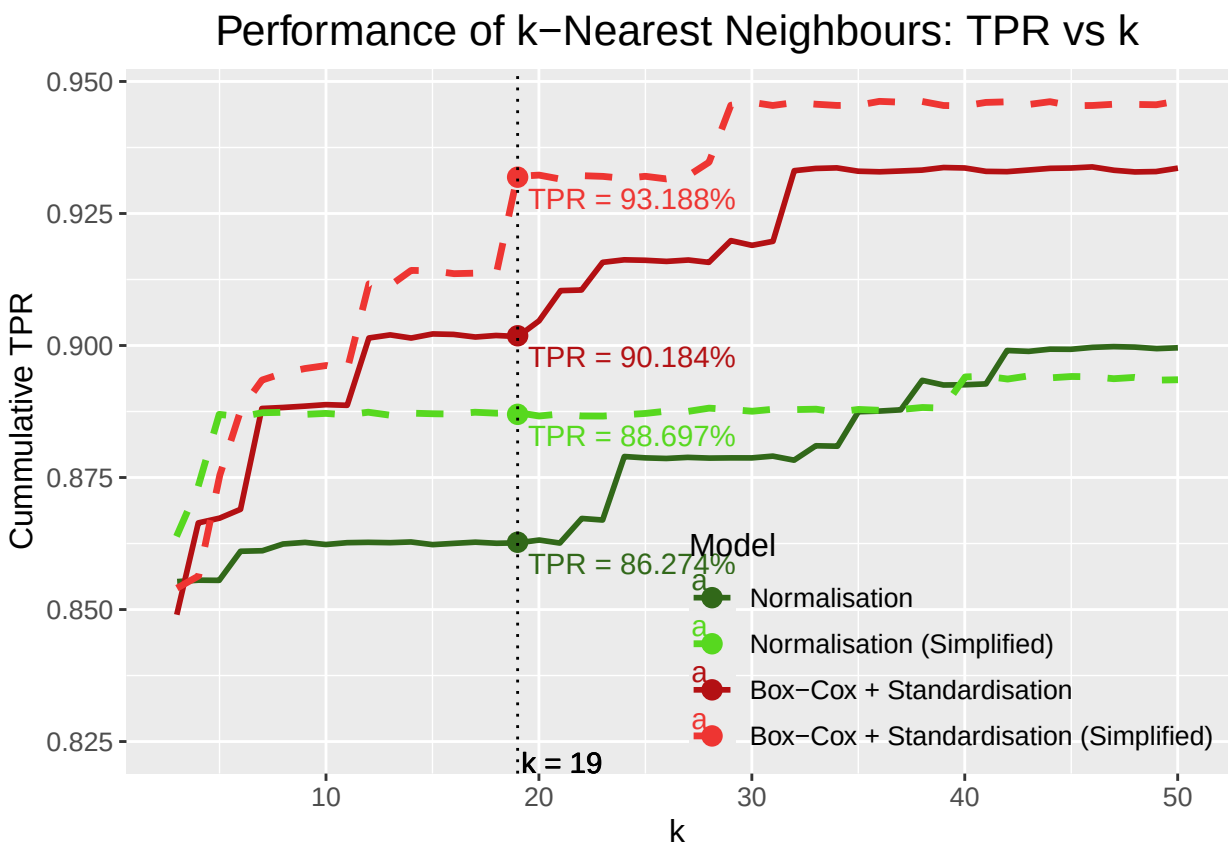
```
(knn.metrics %>%
    ggplot(aes(x = k, y = value, col = transformation)) +
        geom_point(
            data = BEST.MODEL,
            size = 3.1
        ) +
        geom_line(
            data = filter(knn.metrics, metric == 'CTPR', !(transformation %>% endsWith('_sim'))),
            size = 1.0,
            position=position_jitter(w=0.02, h=0.0005)
        ) +
        geom_line(
            data = filter(knn.metrics, metric == 'CTPR', transformation %>% endsWith('_sim')),
            size = 1.2,
            linetype = 'dashed',
            position=position_jitter(w=0.02, h=0.0005)
        ) +
        labs(
            x = 'k',
            y = 'Cummulative TPR',
            col = 'Model'
        ) +
        # scale_alpha_manual(
        #     labels = c('TPR', 'Cummulative TPR'),
        #     breaks = c('TPR', 'CTPR'),
        #     values = c(0.6, 1.0)
        # ) +
        # scale_size_manual(
        #     breaks = c('TPR', 'CTPR'),
        #     values = c(0.3, 1.2)
        # ) +
        scale_color_manual(
            breaks = c(
                'normalised', 'normalised_sim',
                # 'scaled', 'scaled_sim'
                'scaled_boxcox', 'scaled_boxcox_sim'
            ),
            labels = c(
                'Normalisation', 'Normalisation (Simplified)',
                # 'Standardisation', 'Standardisation (Simplified)'
                'Box-Cox + Standardisation', 'Box-Cox + Standardisation (Simplified)'
            ),
            values=c(
                '#316819', '#57d81f',
                # '#d76f24', '#eec666',
                '#b31013', '#ee3532'
            )
        ) +
        geom_text(
            data = BEST.MODEL,
            aes(
                x = k,
```

```
            y = value,
            col = transformation,
            label = paste0('TPR = ', (value * 100) %>% round(3), '%'),
        ),
        hjust = -0.05,
        vjust = 1.5,
        text=element_text(size=11)
    ) +
    geom_vline(
        data = BEST.MODEL,
        aes(xintercept = k),
        col = 'black',
        linetype = 'dotted'
    ) +
    geom_text(
        data = BEST.MODEL,
        aes(
            x = k,
            y = 0.825,
            label = paste0('k = ', k)
        ),
        col = 'black',
        hjust = -0.05,
        vjust = 1.5,
        text=element_text(size=11)
    ) +
    theme(
        legend.position = 'inside',
        # legend.direction = 'horizontal',
        legend.box = 'horizontal',
        legend.box.just = 'right',
        legend.justification = c('right', 'bottom'),
        legend.background = element_blank()
    )) %>%
    .save_and_display(
        'Performance of k-Nearest Neighbours: TPR vs k',
        '../figures/31.kNN.pdf'
    )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning in geom_text(data = BEST.MODEL, aes(x = k, y = value, col =
## transformation, : Ignoring unknown parameters: `text`
```

```
## Warning in geom_text(data = BEST.MODEL, aes(x = k, y = 0.825, label = paste0("k
## = ", : Ignoring unknown parameters: `text`
```



Logistic Regression

```
df_lr <- df
df_lr_scaled <- df %>%
    .mutate_boxcox() %>%
    mutate(across(all_of(NUMERICAL_VARAIBLES), scale))
```

```
model <- glm(disease ~ ., df_lr, family = binomial(link = "logit"))
model_scaled <- glm(disease ~ ., df_lr_scaled, family = binomial(link = "logit"))

p.value.model <- coef(summary(model))[,'Pr(>|z|)']
p.value.model_scaled <- coef(summary(model_scaled))[,'Pr(>|z|)']

cbind(
    p.value.model,
    p.value.model_scaled
)
```

```
##                p.value.model p.value.model_scaled
## (Intercept)      5.660211e-01         6.331983e-01
## age              9.703212e-01         9.660706e-01
## sex1             1.039243e-02         8.503846e-03
## chest.pain1      4.852129e-02         4.366117e-02
## chest.pain2      7.226575e-05         7.123761e-05
## chest.pain3      3.277199e-03         3.075005e-03
## bp               1.254284e-01         1.213443e-01
```

```
## chol             2.096798e-01         1.504399e-01
## fbs1             7.500911e-01         7.367239e-01
## rest.ecg1        7.372820e-02         7.849961e-02
## rest.ecg2        9.289810e-01         8.992408e-01
## heart.rate       2.494968e-02         2.416303e-02
## angina1          6.793420e-02         7.158427e-02
## st.depression    3.859133e-03         3.989920e-03
## vessels          6.099984e-05         6.011475e-05
## blood.disorder2  7.846258e-01         7.880265e-01
## blood.disorder3  9.884810e-02         9.543589e-02
```

```
lr.pipe <- function(params, prob=TRUE, excludes=character(0), tree.parms = list()) {
    if (!prob) { throw('For Logistic Regression, prob must be TRUE!') }

    function(train, test, ...) {
        train <- train %>% select(-all_of(excludes))
        test  <- test  %>% select(-all_of(excludes))

        model <- glm(disease ~ ., train,
                     family = binomial(link = 'logit'))

        list(
            result = model %>% predict(newdata = test, type = 'response'),
            model = model
        )
    }
}

lr.model   <- \(...) lr.pipe()
lr.model.1 <- \(...) lr.pipe(excludes=c('fbs', 'bp', 'chol'))
lr.model.2 <- \(...) lr.pipe(excludes=c('fbs', 'bp', 'chol'))

lr.preds   <- .predict.all(list(1), lr.model,   df_lr)
lr.preds.1 <- .predict.all(list(1), lr.model.1, df_lr)
lr.preds.2 <- .predict.all(list(1), lr.model.2, df_lr_scaled)

lr.preds
```

```
## [[1]]
## A prediction instance
##   with 298 data points
```

```
lr.tprs   <- lr.preds   %>% lapply(performance, 'tpr', 'fpr')
lr.tprs.1 <- lr.preds.1 %>% lapply(performance, 'tpr', 'fpr')
lr.tprs.2 <- lr.preds.2 %>% lapply(performance, 'tpr', 'fpr')

lr.auc   <- lr.preds   %>% lapply(performance, 'auc')
lr.auc.1 <- lr.preds.1 %>% lapply(performance, 'auc')
lr.auc.2 <- lr.preds.2 %>% lapply(performance, 'auc')

.lr.tprs.df <- function(tprs, name=NULL) {
    x <- tprs[[1]]@x.values[[1]]
    y <- tprs[[1]]@y.values[[1]]

    data.frame(
```

```
        # cutoff = x,
        FPR = x, TPR = y, name = name
    )
}

.lr.aucs.df <- \(aucs, name=NULL) data.frame(AUC = aucs[[1]]@y.values[[1]], name = name)

lr.metrics <- .lr.tprs.df(lr.tprs,   name='full') %>%
    # full_join(.lr.tprs.df(lr.tprs.1, name='simp')) %>%
    full_join(.lr.tprs.df(lr.tprs.2, name='sim_scaled'))
```

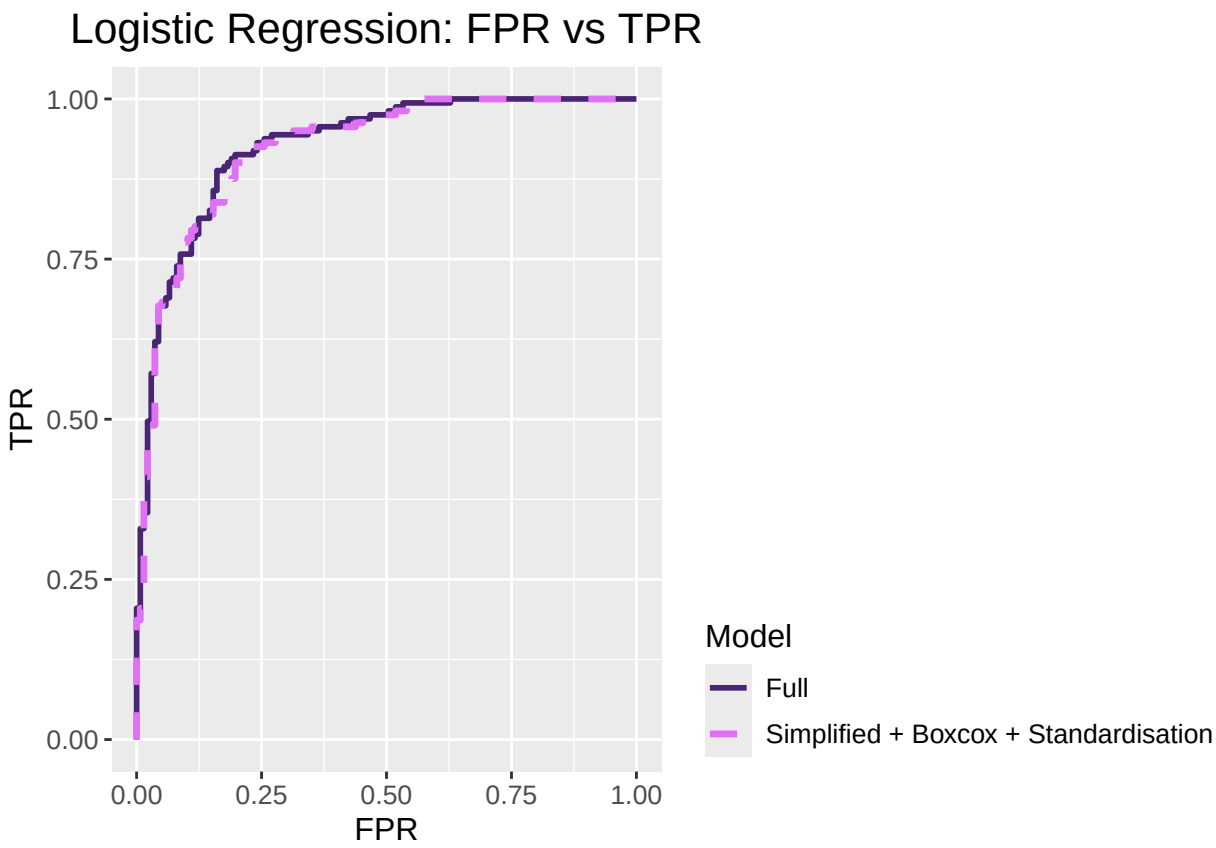
```
## Joining with `by = join_by(FPR, TPR, name)`
```

```
lr.metrics.aucs <- .lr.aucs.df(lr.auc,   name='full') %>%
    # full_join(.lr.aucs.df(lr.auc.1, name='simp')) %>%
    full_join(.lr.aucs.df(lr.auc.2, name='sim_scaled'))
```

```
## Joining with `by = join_by(AUC, name)`
```

```
(lr.metrics %>%
    ggplot(aes(x = FPR, y = TPR, col = name)) +
        geom_line(
            data = filter(lr.metrics, name == 'full'),
            size = 1.0
        ) +
        geom_line(
            data = filter(lr.metrics, name == 'sim_scaled'),
            size = 1.2,
            linetype = 'dashed'
        ) +
        labs(
            # x = 'Cutoff',
            x = 'FPR',
            y = 'TPR',
            col = 'Model'
        ) +
        scale_color_manual(
            breaks = c('full', 'simp', 'sim_scaled'),
            labels = c('Full', 'Simplified', 'Simplified + Boxcox + Standardisation'),
            values=c('#4a2574', '#ffffff', '#de72f3')
        ) +
        theme(
            # legend.position = 'inside',
            # legend.direction = 'horizontal',
            legend.box = 'horizontal',
            legend.box.just = 'right',
            legend.justification = c('left', 'bottom'),
            legend.background = element_blank()
        )

) %>%
    .save_and_display(
        'Logistic Regression: FPR vs TPR',
        '../figures/33.LogisticRegression.pdf'
    )
```




```
lr.best.model <- lr.model()(df_lr_scaled, df_lr_scaled)$model
summary(lr.best.model)
```

```
##
## Call:
## glm(formula = disease ~ ., family = binomial(link = "logit"),
##     data = train)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       0.439582   0.921112   0.477  0.63320
## age               0.009103   0.214013   0.043  0.96607
## sex1             -1.312614   0.498831  -2.631  0.00850 **
## chest.pain1       1.114478   0.552452   2.017  0.04366 *
## chest.pain2       1.858435   0.467871   3.972 7.12e-05 ***
## chest.pain3       1.879491   0.634933   2.960  0.00308 **
## bp                0.287848   0.185809   1.549  0.12134
## chol              0.298888   0.207852   1.438  0.15044
## fbs1              0.186405   0.554455   0.336  0.73672
## rest.ecg1         0.652169   0.370665   1.759  0.07850 .
## rest.ecg2        -0.298933   2.360858  -0.127  0.89924
## heart.rate        0.533273   0.236534   2.255  0.02416 *
## angina1          -0.761245   0.422502  -1.802  0.07158 .
## st.depression    -0.688580   0.239177  -2.879  0.00399 **
## vessels          -0.799324   0.199216  -4.012 6.01e-05 ***
## blood.disorder2   0.209701   0.779923   0.269  0.78803
## blood.disorder3  -1.284189   0.770177  -1.667  0.09544 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 411.18  on 297  degrees of freedom
## Residual deviance: 204.35  on 281  degrees of freedom
## AIC: 238.35
##
## Number of Fisher Scoring iterations: 6
```

```
coef(summary(lr.best.model))[,c('Estimate', 'Pr(>|z|)')] %>% data.frame() %>%
    mutate(abs_est = -abs(Estimate)) %>%
    arrange(order_by = abs_est) %>%
    select(-abs_est) %>%
    mutate(Estimate = round(Estimate, 2))
```

```
##                 Estimate     Pr...z..
## chest.pain3         1.88 3.075005e-03
## chest.pain2         1.86 7.123761e-05
## sex1               -1.31 8.503846e-03
## blood.disorder3    -1.28 9.543589e-02
## chest.pain1         1.11 4.366117e-02
## vessels            -0.80 6.011475e-05
## angina1            -0.76 7.158427e-02
## st.depression      -0.69 3.989920e-03
## rest.ecg1           0.65 7.849961e-02
## heart.rate          0.53 2.416303e-02
## (Intercept)         0.44 6.331983e-01
## rest.ecg2          -0.30 8.992408e-01
## chol                0.30 1.504399e-01
## bp                  0.29 1.213443e-01
## blood.disorder2     0.21 7.880265e-01
## fbs1                0.19 7.367239e-01
## age                 0.01 9.660706e-01
```

```
# %>%
#     knitr::kable(format = 'latex') %>%
#     writeLines()
```

Decision Tree

```
tree.pipe <- function(params, prob=FALSE, excludes=character(0), tree.parms = list()) {
    function(train, test, ...) {
        train <- train %>% select(-all_of(excludes))
        test  <- test  %>% select(-all_of(excludes))

        model <- rpart(
            disease ~ ., train,
            control = rpart.control %>% do.call(params),
            parms = tree.parms
        )

        if (prob)
```

```
            list(
                result = model %>% predict(newdata = test, type = 'vector'),
                model = model
            )
        else
            list(
                result = model %>% predict(newdata = test, type = 'class') %>%
                    as.character() %>% as.numeric(),
                model = model
            )
    }
}

tree.model.info     <- \(params, prob=FALSE) tree.pipe(
    params, prob, tree.parms = list(split = 'information')
)
tree.model.info_sim <- \(params, prob=FALSE) tree.pipe(
    params, prob, tree.parms = list(split = 'information'), excludes=c('fbs', 'bp', 'chol')
)
tree.model.gini     <- \(params, prob=FALSE) tree.pipe(
    params, prob, tree.parms = list(split = 'gini')
)
tree.model.gini_sim <- \(params, prob=FALSE) tree.pipe(
    params, prob, tree.parms = list(split = 'gini'), excludes=c('fbs', 'bp', 'chol')
)

tree.params.df <- expand.grid(minsplit = seq(5, 100, 5))
# tree.params.df <- expand.grid(maxdepth = seq(2, 20, 1))
tree.params    <- tree.params.df %>% as.list() %>% transpose()

tree.preds.info     <- .predict.all.cv(tree.params, tree.model.info,     df)
tree.preds.info_sim <- .predict.all.cv(tree.params, tree.model.info_sim, df)
tree.preds.gini     <- .predict.all.cv(tree.params, tree.model.gini,     df)
tree.preds.gini_sim <- .predict.all.cv(tree.params, tree.model.gini_sim, df)

tree.tprs.info      <- tree.preds.info     %>% .single.metric('tpr')
tree.tprs.info_sim  <- tree.preds.info_sim %>% .single.metric('tpr')
tree.tprs.gini      <- tree.preds.gini     %>% .single.metric('tpr')
tree.tprs.gini_sim  <- tree.preds.gini_sim %>% .single.metric('tpr')

tree.fprs.info      <- tree.preds.info     %>% .single.metric('fpr')
tree.fprs.info_sim  <- tree.preds.info_sim %>% .single.metric('fpr')
tree.fprs.gini      <- tree.preds.gini     %>% .single.metric('fpr')
tree.fprs.gini_sim  <- tree.preds.gini_sim %>% .single.metric('fpr')

tree.metrics <- data.frame(
    split = tree.params.df$minsplit,
    # depth = tree.params.df$maxdepth,
     TPR.info     =        tree.tprs.info,
    CTPR.info     = cummax(tree.tprs.info),
     TPR.info_sim =        tree.tprs.info_sim,
    CTPR.info_sim = cummax(tree.tprs.info_sim),
     TPR.gini     =        tree.tprs.gini,
```

```
    CTPR.gini     = cummax(tree.tprs.gini),
     TPR.gini_sim =        tree.tprs.gini_sim,
    CTPR.gini_sim = cummax(tree.tprs.gini_sim),
     FPR.info     =        tree.fprs.info,
    CFPR.info     = cummax(tree.fprs.info),
     FPR.info_sim =        tree.fprs.info_sim,
    CFPR.info_sim = cummax(tree.fprs.info_sim),
     FPR.gini     =        tree.fprs.gini,
    CFPR.gini     = cummax(tree.fprs.gini),
     FPR.gini_sim =        tree.fprs.gini_sim,
    CFPR.gini_sim = cummax(tree.fprs.gini_sim)
) %>%
    pivot_longer(
        cols = starts_with(c('TPR', 'CTPR', 'FPR', 'CFPR')),
        names_to = 'metric',
        values_to = 'value'
    ) %>%
    mutate(transformation = gsub('.*\\.', '', metric)) %>%
    mutate(metric = gsub('\\..*', '', metric)) %>%
    mutate(generic_metric = metric %>% substr(nchar(.) - 2, nchar(.)))

BEST.SPLIT <- c(30, 45, 60)
BEST.MODEL <- tree.metrics %>% filter(
    # transformation == 'scaled_boxcox_sim',
    metric %in% c('CTPR', 'CFPR'), split %in% BEST.SPLIT, transformation != 'gini_sim'
)
BEST.MODEL
```

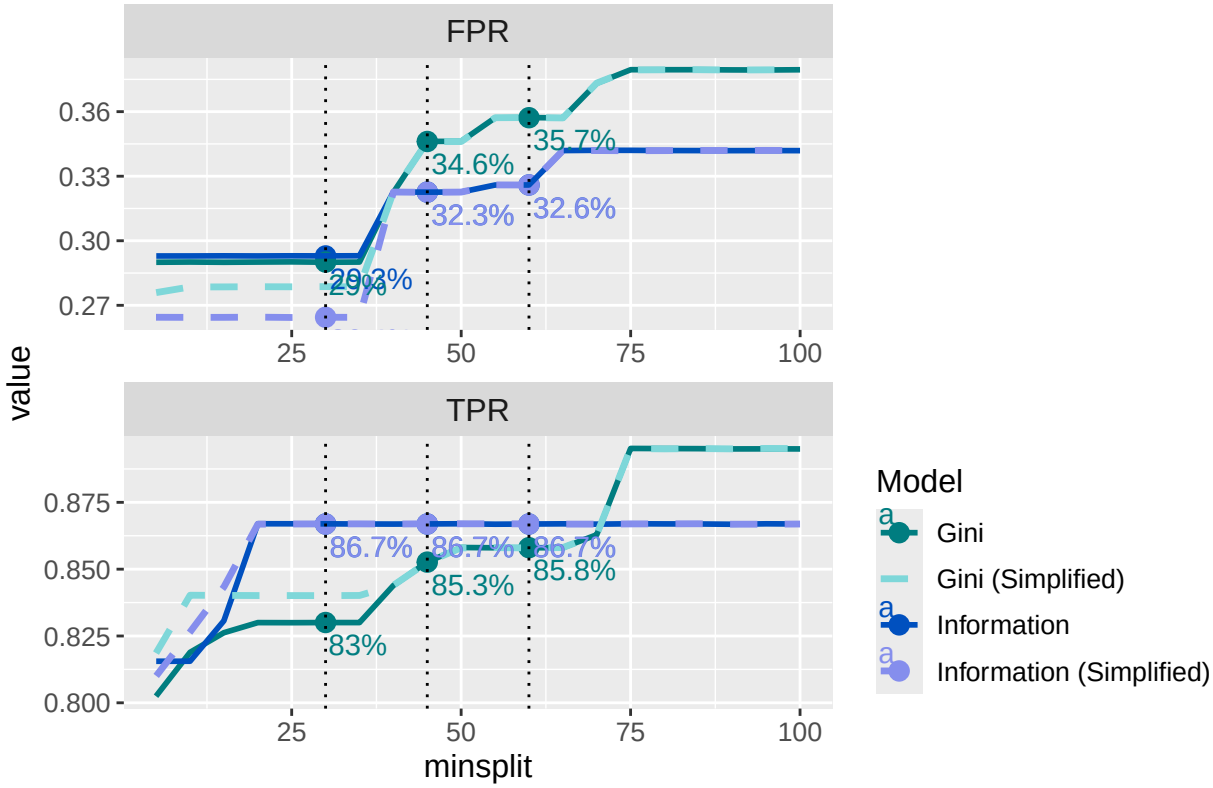
```
## # A tibble: 18 x 5
##    split metric value transformation generic_metric
##    <dbl> <chr>  <dbl> <chr>          <chr>
##  1    30 CTPR   0.867 info           TPR
##  2    30 CTPR   0.867 info_sim       TPR
##  3    30 CTPR   0.830 gini           TPR
##  4    30 CFPR   0.293 info           FPR
##  5    30 CFPR   0.264 info_sim       FPR
##  6    30 CFPR   0.290 gini           FPR
##  7    45 CTPR   0.867 info           TPR
##  8    45 CTPR   0.867 info_sim       TPR
##  9    45 CTPR   0.853 gini           TPR
## 10    45 CFPR   0.323 info           FPR
## 11    45 CFPR   0.323 info_sim       FPR
## 12    45 CFPR   0.346 gini           FPR
## 13    60 CTPR   0.867 info           TPR
## 14    60 CTPR   0.867 info_sim       TPR
## 15    60 CTPR   0.858 gini           TPR
## 16    60 CFPR   0.326 info           FPR
## 17    60 CFPR   0.326 info_sim       FPR
## 18    60 CFPR   0.357 gini           FPR
```

```
(tree.metrics %>%
    ggplot(aes(x = split, y = value, col = transformation)) +
        facet_wrap(generic_metric ~ ., scales = 'free', nrow=2) +
        geom_point(
```

```
        data = BEST.MODEL,
        size = 3.1
    ) +
    geom_line(
        data = filter(tree.metrics, startsWith(metric, 'C'), !(transformation %>% endsWith('_sim'))),
        size = 1.0,
        position=position_jitter(w=0.02, h=0.0001)
    ) +
    geom_line(
        data = filter(tree.metrics, startsWith(metric, 'C'), transformation %>% endsWith('_sim')),
        size = 1.2,
        linetype = 'dashed',
        position=position_jitter(w=0.02, h=0.0001)
    ) +
    labs(
        x = 'minsplit',
        col = 'Model'
    ) +
    # scale_alpha_manual(
    #     labels = c('TPR', 'Cummulative TPR'),
    #     breaks = c('TPR', 'CTPR'),
    #     values = c(0.6, 1.0)
    # ) +
    # scale_size_manual(
    #     breaks = c('TPR', 'CTPR'),
    #     values = c(0.3, 1.2)
    # ) +
    scale_color_manual(
        breaks = c(
            'gini',
            'gini_sim',
            'info',
            'info_sim'
        ),
        labels = c(
            'Gini',
            'Gini (Simplified)',
            'Information',
            'Information (Simplified)'
        ),
        values=c(
            '#007d80',
            '#7ed7d9',
            '#0050bf',
            '#858eed'
        )
    ) +
    geom_text(
        data = BEST.MODEL,
        aes(
            x = split,
            y = value,
            col = transformation,
```

```
            label = paste0((value * 100) %>% round(1), '%'),
        ),
        hjust = -0.05,
        vjust =  1.55,
        text=element_text(size=11)
    ) +
    geom_vline(
        data = BEST.MODEL,
        aes(xintercept = split),
        col = 'black',
        linetype = 'dotted'
    ) +
    # geom_text(
    #     data = BEST.MODEL,
    #     aes(
    #         x = split,
    #         y = 0.847,
    #         label = paste0('minsplit = ', split)
    #     ),
    #     col = 'black',
    #     hjust = 1.15,
    #     vjust = 0.5,
    #     text=element_text(size=11)
    # ) +
    theme(
        # legend.position = 'inside',
        # legend.direction = 'horizontal',
        legend.box = 'horizontal',
        legend.box.just = 'right',
        legend.justification = c('right', 'bottom'),
        legend.background = element_blank()
    )) %>%
    .save_and_display(
        'Performance of Decision Tree: (FPR, TPR) vs minsplit',
        width = 960, height = 960,
        '../figures/32.DecisionTree.pdf'
    )
```

```
## Warning in geom_text(data = BEST.MODEL, aes(x = split, y = value, col =
## transformation, : Ignoring unknown parameters: `text`
```

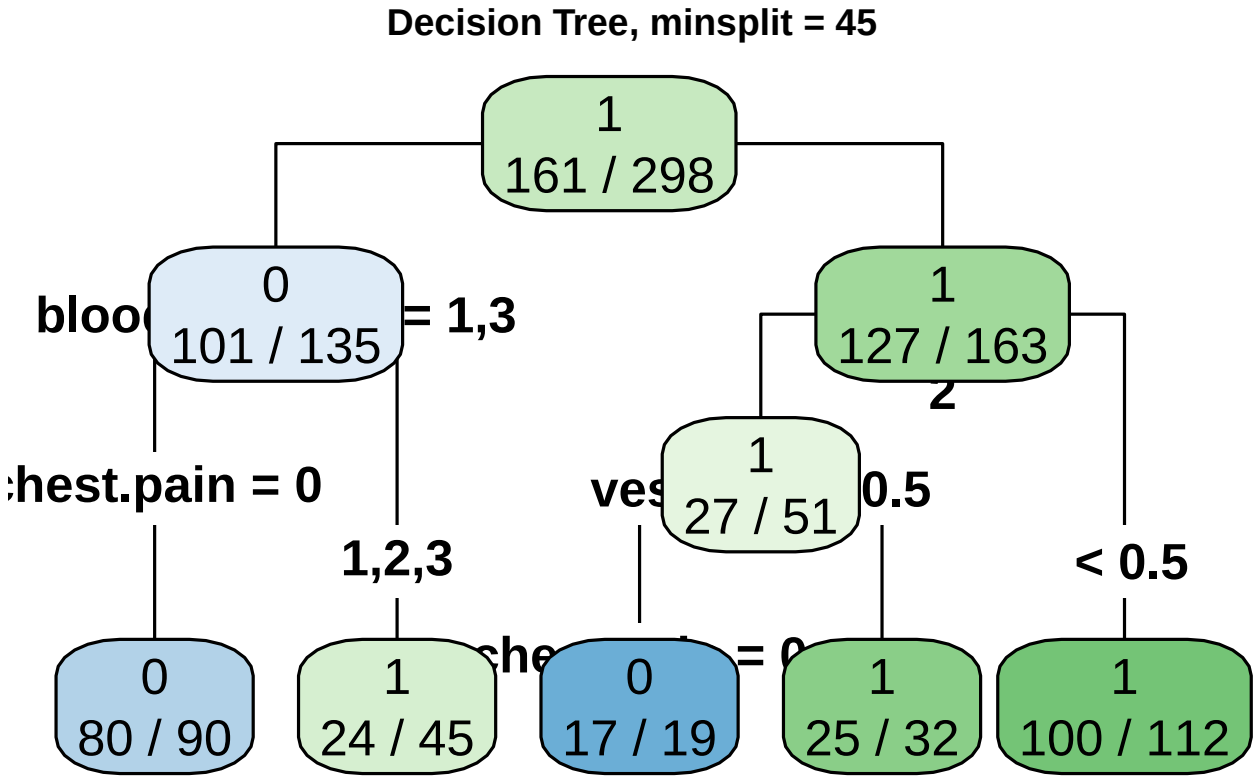



```
tree.metrics %>%
    filter(metric == 'TPR') %>% slice_max(value, n = 1)
```

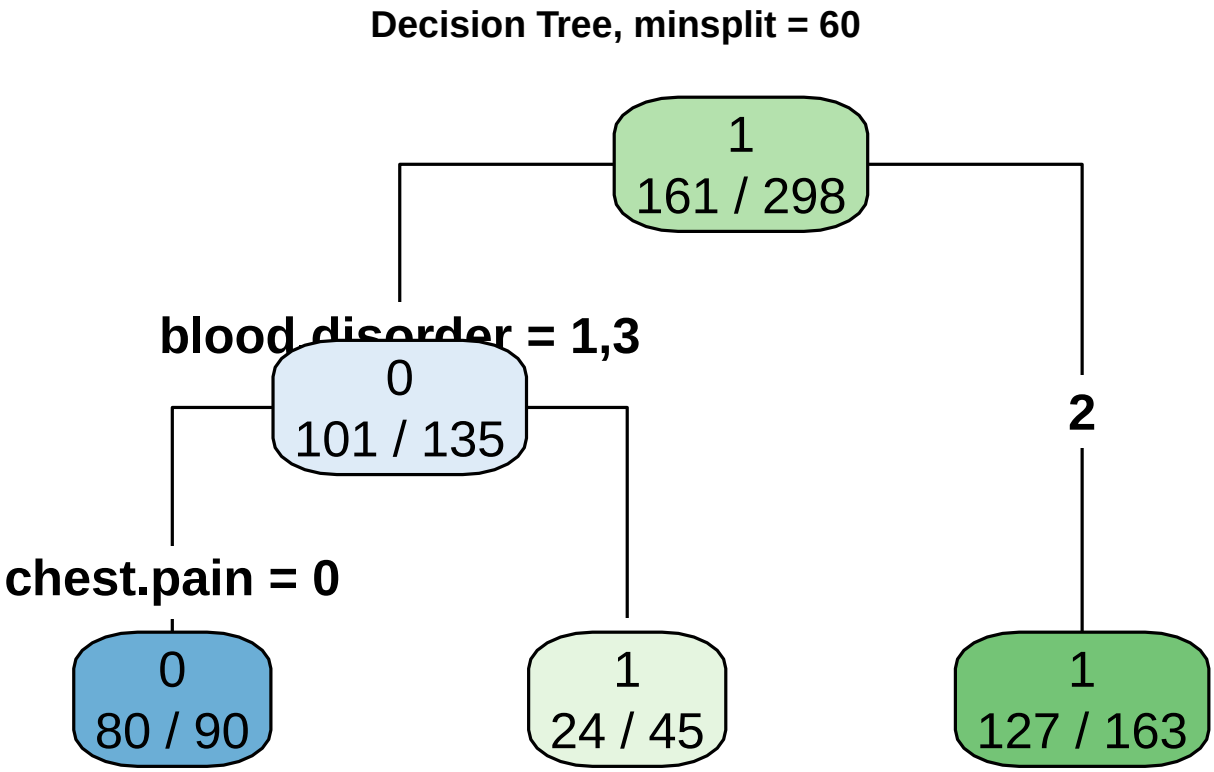
```
## # A tibble: 10 x 5
##    split metric value transformation generic_metric
##    <dbl> <chr>  <dbl> <chr>          <chr>
##  1    75 TPR    0.895 gini           TPR
##  2    75 TPR    0.895 gini_sim       TPR
##  3    80 TPR    0.895 gini           TPR
##  4    80 TPR    0.895 gini_sim       TPR
##  5    85 TPR    0.895 gini           TPR
##  6    85 TPR    0.895 gini_sim       TPR
##  7    90 TPR    0.895 gini           TPR
##  8    90 TPR    0.895 gini_sim       TPR
##  9    95 TPR    0.895 gini           TPR
## 10    95 TPR    0.895 gini_sim       TPR
```

```
tree.best.model.1 <- tree.model.gini_sim(params = list(minsplit = 50))(df, df)$model
tree.best.model.2 <- tree.model.gini_sim(params = list(minsplit = 60))(df, df)$model

# pdf('../figures/32.DecisionTree-45.pdf')
tree.best.model.1 %>%
    rpart.plot(
        type = 4, extra = 2, roundint=F,
        # width = 960, height = 640,
        xpd = T, cex = 1.6,
        main = paste0('Decision Tree, minsplit = 45')
    )
```




```
# pdf('../figures/32.DecisionTree-60.pdf')
tree.best.model.2 %>%
    rpart.plot(
        type = 4, extra = 2, roundint=F,
        # width = 960, height = 640,
        xpd = T, cex = 1.6,
        main = paste0('Decision Tree, minsplit = 60')
    )
```




```
# dev.off()
```

Choosing the best model

```
final.knn.model <- knn3(
    disease ~
        sex + chest.pain + rest.ecg + heart.rate +
        angina + st.depression + vessels + blood.disorder,
    df_knn_scaled_boxcox,
    k = 19
)

final.lr.model <- glm(
    disease ~ ., df,
    family = 'binomial'
)

final.tree.small.model <- rpart(
    disease ~ ., df,
    control = rpart.control(minsplit = 30)
)

final.tree.medium.model <- rpart(
    disease ~ ., df,
    control = rpart.control(minsplit = 45)
)
```

```
.predict.final <-
    \(model, data) .all(model, data)

.perf.final <-
    \(pred, measure, x.measure=NULL) performance(pred, measure = measure, x.measure = x.measure)

.final.perfs <- function(preds, y.measure, x.measure = '') {
    seq_along(preds) %>%
        lapply(function(i) {
            perf <- .perf.final %>% do.call(c(
                preds[[i]],
                list(
                    measure = y.measure,
                    x.measure = if (x.measure == '') 'cutoff' else x.measure
                )
            ))

            ys <- perf@y.values[[1]]
            df <- data.frame(matrix(nrow = length(ys), ncol = 0))
            df[y.measure] <- ys
            if (x.measure != '')
                df[x.measure] <- perf@x.values[[1]]
            df['Name'] <- names(preds[i])
            df
        }) %>%
        Reduce(\(df1, df2) full_join(df1, df2), .)
}

final.preds <- list(
    knn = final.knn.model %>%
        predict(df_knn_scaled_boxcox, type = 'prob') %>% (\(x) x[,2]) %>%
        prediction(df_knn_scaled_boxcox$disease),

    lr = final.lr.model %>%
        predict(df, type = 'response') %>%
        prediction(df_knn_scaled_boxcox$disease),

    tree.small = final.tree.small.model %>%
        predict(df, type = 'prob') %>% (\(x) x[,2]) %>%
        prediction(df_knn_scaled_boxcox$disease),

    tree.medium = final.tree.medium.model %>%
        predict(df, type = 'prob') %>% (\(x) x[,2]) %>%
        prediction(df_knn_scaled_boxcox$disease)
)
```

```
final.ROC.df <- final.preds %>% .final.perfs('tpr', 'fpr')
```

```
## Joining with `by = join_by(tpr, fpr, Name)`
## Joining with `by = join_by(tpr, fpr, Name)`
## Joining with `by = join_by(tpr, fpr, Name)`
```

```
final.ROC.df
```

```
##            tpr        fpr        Name
```

```
## 1   0.00000000 0.00000000         knn
## 2   0.19254658 0.00000000         knn
## 3   0.36645963 0.00000000         knn
## 4   0.54037267 0.02919708         knn
## 5   0.55900621 0.02919708         knn
## 6   0.60248447 0.05109489         knn
## 7   0.65838509 0.07299270         knn
## 8   0.72049689 0.10948905         knn
## 9   0.72670807 0.10948905         knn
## 10  0.81366460 0.13138686         knn
## 11  0.81987578 0.13138686         knn
## 12  0.86956522 0.18248175         knn
## 13  0.87577640 0.18248175         knn
## 14  0.89440994 0.22627737         knn
## 15  0.91304348 0.24817518         knn
## 16  0.95031056 0.31386861         knn
## 17  0.97515528 0.32846715         knn
## 18  0.97515528 0.33576642         knn
## 19  0.97515528 0.37956204         knn
## 20  0.98136646 0.42335766         knn
## 21  0.98136646 0.45255474         knn
## 22  0.98136646 0.46715328         knn
## 23  0.98136646 0.53284672         knn
## 24  0.99378882 0.60583942         knn
## 25  1.00000000 0.81751825         knn
## 26  1.00000000 0.82481752         knn
## 27  1.00000000 1.00000000         knn
## 28  0.00000000 0.00000000          lr
## 29  0.00621118 0.00000000          lr
## 30  0.01242236 0.00000000          lr
## 31  0.01863354 0.00000000          lr
## 32  0.02484472 0.00000000          lr
## 33  0.03105590 0.00000000          lr
## 34  0.03726708 0.00000000          lr
## 35  0.04347826 0.00000000          lr
## 36  0.04968944 0.00000000          lr
## 37  0.05590062 0.00000000          lr
## 38  0.06211180 0.00000000          lr
## 39  0.06832298 0.00000000          lr
## 40  0.07453416 0.00000000          lr
## 41  0.08074534 0.00000000          lr
## 42  0.08695652 0.00000000          lr
## 43  0.09316770 0.00000000          lr
## 44  0.09937888 0.00000000          lr
## 45  0.10559006 0.00000000          lr
## 46  0.11180124 0.00000000          lr
## 47  0.11801242 0.00000000          lr
## 48  0.12422360 0.00000000          lr
## 49  0.13043478 0.00000000          lr
## 50  0.13664596 0.00000000          lr
## 51  0.14285714 0.00000000          lr
## 52  0.14906832 0.00000000          lr
## 53  0.15527950 0.00000000          lr
## 54  0.16149068 0.00000000          lr
```

```
## 55  0.16770186 0.00000000          lr
## 56  0.17391304 0.00000000          lr
## 57  0.18012422 0.00000000          lr
## 58  0.18633540 0.00000000          lr
## 59  0.19254658 0.00000000          lr
## 60  0.19875776 0.00000000          lr
## 61  0.20496894 0.00000000          lr
## 62  0.20496894 0.00729927          lr
## 63  0.21118012 0.00729927          lr
## 64  0.21739130 0.00729927          lr
## 65  0.22360248 0.00729927          lr
## 66  0.22981366 0.00729927          lr
## 67  0.23602484 0.00729927          lr
## 68  0.24223602 0.00729927          lr
## 69  0.24844720 0.00729927          lr
## 70  0.25465839 0.00729927          lr
## 71  0.26086957 0.00729927          lr
## 72  0.26708075 0.00729927          lr
## 73  0.27329193 0.00729927          lr
## 74  0.27950311 0.00729927          lr
## 75  0.28571429 0.00729927          lr
## 76  0.29192547 0.00729927          lr
## 77  0.29813665 0.00729927          lr
## 78  0.30434783 0.00729927          lr
## 79  0.31055901 0.00729927          lr
## 80  0.31677019 0.00729927          lr
## 81  0.32298137 0.00729927          lr
## 82  0.32919255 0.00729927          lr
## 83  0.32919255 0.01459854          lr
## 84  0.33540373 0.01459854          lr
## 85  0.34161491 0.01459854          lr
## 86  0.34782609 0.01459854          lr
## 87  0.35403727 0.01459854          lr
## 88  0.35403727 0.02189781          lr
## 89  0.36024845 0.02189781          lr
## 90  0.36645963 0.02189781          lr
## 91  0.37267081 0.02189781          lr
## 92  0.37888199 0.02189781          lr
## 93  0.38509317 0.02189781          lr
## 94  0.39130435 0.02189781          lr
## 95  0.39751553 0.02189781          lr
## 96  0.40372671 0.02189781          lr
## 97  0.40993789 0.02189781          lr
## 98  0.41614907 0.02189781          lr
## 99  0.42236025 0.02189781          lr
## 100 0.42857143 0.02189781          lr
## 101 0.43478261 0.02189781          lr
## 102 0.44099379 0.02189781          lr
## 103 0.44720497 0.02189781          lr
## 104 0.45341615 0.02189781          lr
## 105 0.45962733 0.02189781          lr
## 106 0.46583851 0.02189781          lr
## 107 0.47204969 0.02189781          lr
## 108 0.47826087 0.02189781          lr
```

```
## 109 0.48447205 0.02189781          lr
## 110 0.49068323 0.02189781          lr
## 111 0.49689441 0.02189781          lr
## 112 0.49689441 0.02919708          lr
## 113 0.50310559 0.02919708          lr
## 114 0.50931677 0.02919708          lr
## 115 0.51552795 0.02919708          lr
## 116 0.52173913 0.02919708          lr
## 117 0.52795031 0.02919708          lr
## 118 0.53416149 0.02919708          lr
## 119 0.54037267 0.02919708          lr
## 120 0.54658385 0.02919708          lr
## 121 0.55279503 0.02919708          lr
## 122 0.55900621 0.02919708          lr
## 123 0.56521739 0.02919708          lr
## 124 0.57142857 0.02919708          lr
## 125 0.57142857 0.03649635          lr
## 126 0.57763975 0.03649635          lr
## 127 0.58385093 0.03649635          lr
## 128 0.59006211 0.03649635          lr
## 129 0.59627329 0.03649635          lr
## 130 0.60248447 0.03649635          lr
## 131 0.60869565 0.03649635          lr
## 132 0.61490683 0.03649635          lr
## 133 0.62111801 0.03649635          lr
## 134 0.62111801 0.04379562          lr
## 135 0.62732919 0.04379562          lr
## 136 0.63354037 0.04379562          lr
## 137 0.63975155 0.04379562          lr
## 138 0.64596273 0.04379562          lr
## 139 0.65217391 0.04379562          lr
## 140 0.65838509 0.04379562          lr
## 141 0.66459627 0.04379562          lr
## 142 0.67080745 0.04379562          lr
## 143 0.67701863 0.04379562          lr
## 144 0.67701863 0.05109489          lr
## 145 0.67701863 0.05839416          lr
## 146 0.68322981 0.05839416          lr
## 147 0.68944099 0.05839416          lr
## 148 0.68944099 0.06569343          lr
## 149 0.69565217 0.06569343          lr
## 150 0.70186335 0.06569343          lr
## 151 0.70807453 0.06569343          lr
## 152 0.71428571 0.06569343          lr
## 153 0.71428571 0.07299270          lr
## 154 0.72049689 0.07299270          lr
## 155 0.72049689 0.08029197          lr
## 156 0.72670807 0.08029197          lr
## 157 0.73291925 0.08029197          lr
## 158 0.73913043 0.08029197          lr
## 159 0.73913043 0.08759124          lr
## 160 0.74534161 0.08759124          lr
## 161 0.75155280 0.08759124          lr
## 162 0.75776398 0.08759124          lr
```

```
## 163 0.75776398 0.09489051         lr
## 164 0.75776398 0.10218978         lr
## 165 0.75776398 0.10948905         lr
## 166 0.76397516 0.10948905         lr
## 167 0.77018634 0.10948905         lr
## 168 0.77639752 0.10948905         lr
## 169 0.78260870 0.10948905         lr
## 170 0.78260870 0.11678832         lr
## 171 0.78881988 0.11678832         lr
## 172 0.78881988 0.12408759         lr
## 173 0.80124224 0.12408759         lr
## 174 0.80745342 0.12408759         lr
## 175 0.81366460 0.12408759         lr
## 176 0.81366460 0.13138686         lr
## 177 0.81366460 0.13868613         lr
## 178 0.81366460 0.14598540         lr
## 179 0.81987578 0.14598540         lr
## 180 0.82608696 0.14598540         lr
## 181 0.82608696 0.15328467         lr
## 182 0.83229814 0.15328467         lr
## 183 0.83850932 0.15328467         lr
## 184 0.84472050 0.15328467         lr
## 185 0.85093168 0.15328467         lr
## 186 0.85714286 0.15328467         lr
## 187 0.85714286 0.16058394         lr
## 188 0.86335404 0.16058394         lr
## 189 0.86956522 0.16058394         lr
## 190 0.87577640 0.16058394         lr
## 191 0.88198758 0.16058394         lr
## 192 0.88819876 0.16058394         lr
## 193 0.88819876 0.16788321         lr
## 194 0.88819876 0.17518248         lr
## 195 0.89440994 0.17518248         lr
## 196 0.89440994 0.18248175         lr
## 197 0.90062112 0.18248175         lr
## 198 0.90062112 0.18978102         lr
## 199 0.90683230 0.18978102         lr
## 200 0.90683230 0.19708029         lr
## 201 0.91304348 0.19708029         lr
## 202 0.91304348 0.20437956         lr
## 203 0.91304348 0.21167883         lr
## 204 0.91304348 0.21897810         lr
## 205 0.91304348 0.22627737         lr
## 206 0.91304348 0.23357664         lr
## 207 0.91925466 0.23357664         lr
## 208 0.91925466 0.24087591         lr
## 209 0.92546584 0.24087591         lr
## 210 0.93167702 0.24087591         lr
## 211 0.93167702 0.24817518         lr
## 212 0.93167702 0.25547445         lr
## 213 0.93788820 0.25547445         lr
## 214 0.93788820 0.26277372         lr
## 215 0.93788820 0.27007299         lr
## 216 0.94409938 0.27007299         lr
```

```
## 217 0.94409938 0.27737226         lr
## 218 0.94409938 0.28467153         lr
## 219 0.94409938 0.29197080         lr
## 220 0.94409938 0.29927007         lr
## 221 0.94409938 0.30656934         lr
## 222 0.94409938 0.31386861         lr
## 223 0.94409938 0.32116788         lr
## 224 0.94409938 0.32846715         lr
## 225 0.94409938 0.33576642         lr
## 226 0.94409938 0.34306569         lr
## 227 0.95031056 0.34306569         lr
## 228 0.95031056 0.35036496         lr
## 229 0.95031056 0.35766423         lr
## 230 0.95031056 0.36496350         lr
## 231 0.95652174 0.36496350         lr
## 232 0.95652174 0.37226277         lr
## 233 0.95652174 0.37956204         lr
## 234 0.95652174 0.38686131         lr
## 235 0.95652174 0.39416058         lr
## 236 0.95652174 0.40145985         lr
## 237 0.95652174 0.40875912         lr
## 238 0.96273292 0.40875912         lr
## 239 0.96273292 0.41605839         lr
## 240 0.96273292 0.42335766         lr
## 241 0.96894410 0.42335766         lr
## 242 0.96894410 0.43065693         lr
## 243 0.96894410 0.43795620         lr
## 244 0.96894410 0.44525547         lr
## 245 0.96894410 0.45255474         lr
## 246 0.96894410 0.45985401         lr
## 247 0.96894410 0.46715328         lr
## 248 0.97515528 0.46715328         lr
## 249 0.97515528 0.47445255         lr
## 250 0.97515528 0.48175182         lr
## 251 0.97515528 0.48905109         lr
## 252 0.97515528 0.49635036         lr
## 253 0.97515528 0.50364964         lr
## 254 0.98136646 0.50364964         lr
## 255 0.98136646 0.51094891         lr
## 256 0.98136646 0.51824818         lr
## 257 0.98757764 0.51824818         lr
## 258 0.98757764 0.52554745         lr
## 259 0.98757764 0.53284672         lr
## 260 0.99378882 0.53284672         lr
## 261 0.99378882 0.54014599         lr
## 262 0.99378882 0.54744526         lr
## 263 0.99378882 0.55474453         lr
## 264 0.99378882 0.56204380         lr
## 265 0.99378882 0.56934307         lr
## 266 0.99378882 0.57664234         lr
## 267 0.99378882 0.58394161         lr
## 268 0.99378882 0.59124088         lr
## 269 0.99378882 0.59854015         lr
## 270 0.99378882 0.60583942         lr
```

```
## 271 0.99378882 0.61313869         lr
## 272 0.99378882 0.62043796         lr
## 273 0.99378882 0.62773723         lr
## 274 1.00000000 0.62773723         lr
## 275 1.00000000 0.63503650         lr
## 276 1.00000000 0.64233577         lr
## 277 1.00000000 0.64963504         lr
## 278 1.00000000 0.65693431         lr
## 279 1.00000000 0.66423358         lr
## 280 1.00000000 0.67153285         lr
## 281 1.00000000 0.67883212         lr
## 282 1.00000000 0.68613139         lr
## 283 1.00000000 0.69343066         lr
## 284 1.00000000 0.70072993         lr
## 285 1.00000000 0.70802920         lr
## 286 1.00000000 0.71532847         lr
## 287 1.00000000 0.72262774         lr
## 288 1.00000000 0.72992701         lr
## 289 1.00000000 0.73722628         lr
## 290 1.00000000 0.74452555         lr
## 291 1.00000000 0.75182482         lr
## 292 1.00000000 0.75912409         lr
## 293 1.00000000 0.76642336         lr
## 294 1.00000000 0.77372263         lr
## 295 1.00000000 0.78102190         lr
## 296 1.00000000 0.78832117         lr
## 297 1.00000000 0.79562044         lr
## 298 1.00000000 0.80291971         lr
## 299 1.00000000 0.81021898         lr
## 300 1.00000000 0.81751825         lr
## 301 1.00000000 0.82481752         lr
## 302 1.00000000 0.83211679         lr
## 303 1.00000000 0.83941606         lr
## 304 1.00000000 0.84671533         lr
## 305 1.00000000 0.85401460         lr
## 306 1.00000000 0.86131387         lr
## 307 1.00000000 0.86861314         lr
## 308 1.00000000 0.87591241         lr
## 309 1.00000000 0.88321168         lr
## 310 1.00000000 0.89051095         lr
## 311 1.00000000 0.89781022         lr
## 312 1.00000000 0.90510949         lr
## 313 1.00000000 0.91240876         lr
## 314 1.00000000 0.91970803         lr
## 315 1.00000000 0.92700730         lr
## 316 1.00000000 0.93430657         lr
## 317 1.00000000 0.94160584         lr
## 318 1.00000000 0.94890511         lr
## 319 1.00000000 0.95620438         lr
## 320 1.00000000 0.96350365         lr
## 321 1.00000000 0.97080292         lr
## 322 1.00000000 0.97810219         lr
## 323 1.00000000 0.98540146         lr
## 324 1.00000000 0.99270073         lr
```

```
## 325 1.00000000 1.00000000          lr
## 326 0.00000000 0.00000000  tree.small
## 327 0.62111801 0.08759124  tree.small
## 328 0.77639752 0.13868613  tree.small
## 329 0.91304348 0.21897810  tree.small
## 330 0.92546584 0.29197080  tree.small
## 331 0.98757764 0.87591241  tree.small
## 332 1.00000000 1.00000000  tree.small
## 333 0.00000000 0.00000000 tree.medium
## 334 0.62111801 0.08759124 tree.medium
## 335 0.77639752 0.13868613 tree.medium
## 336 0.89440994 0.19708029 tree.medium
## 337 0.92546584 0.29197080 tree.medium
## 338 0.98757764 0.87591241 tree.medium
## 339 1.00000000 1.00000000 tree.medium
```

```
final.AUC.df <- final.preds %>% .final.perfs('auc')
```

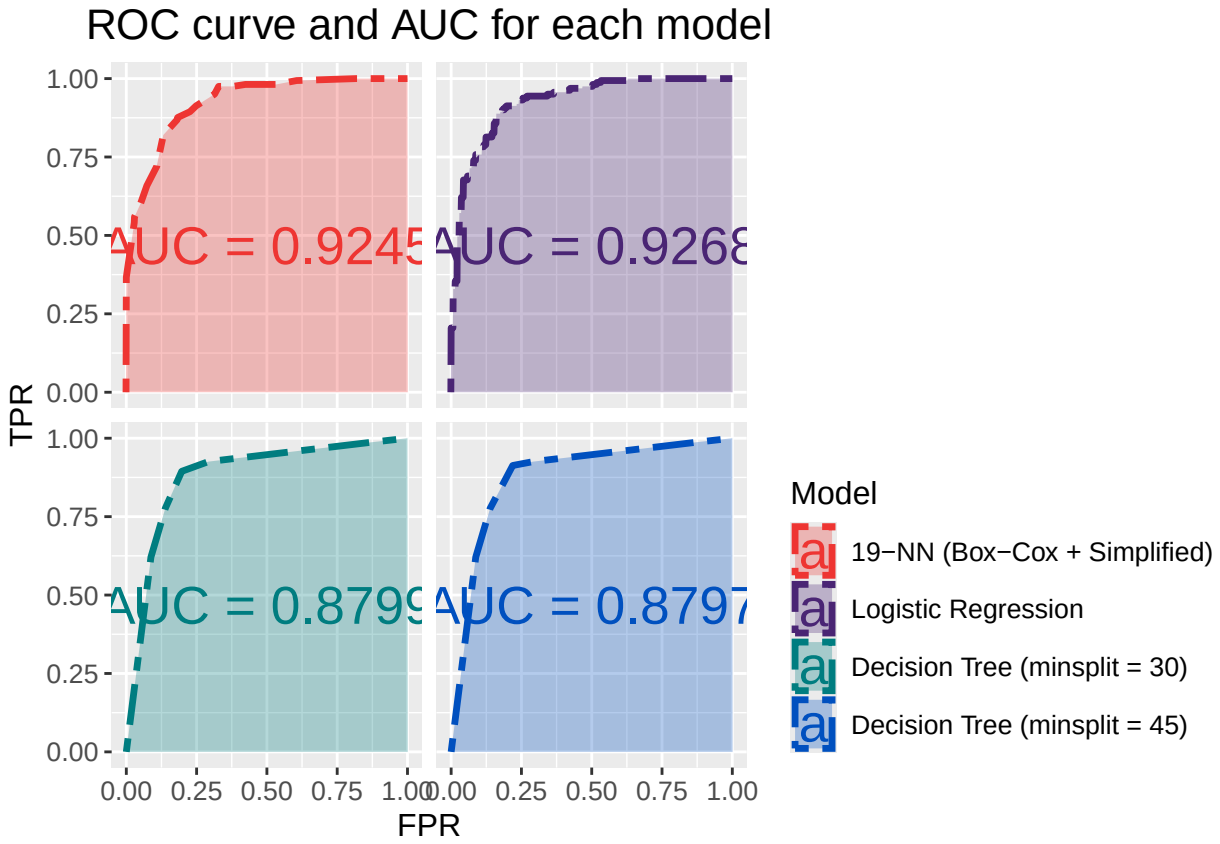
```
## Joining with `by = join_by(auc, Name)`
## Joining with `by = join_by(auc, Name)`
## Joining with `by = join_by(auc, Name)`
```

```
final.AUC.df
```

```
##         auc        Name
## 1 0.9244684         knn
## 2 0.9267806          lr
## 3 0.8796981  tree.small
## 4 0.8799021 tree.medium
```

```
(ggplot() +
    facet_wrap(~ Name) +
    geom_area(
        data = final.ROC.df,
        mapping = aes(x = fpr, y = tpr, col = Name, fill = Name),
        size = 1.2, alpha = .3,
        linetype = 'twodash'
    ) +
    geom_text(
        data = final.AUC.df,
        mapping = aes(
            x = 0.5,
            y = 0.5,
            label = paste0('AUC = ', auc %>% round(4)),
            col = Name
        ),
        size = 7,
        hjust =  .5,
        vjust =  .75
    ) +
        labs(
            x = 'FPR',
            y = 'TPR',
            color = 'Model',
            fill = 'Model'
        ) +
```

```
        scale_color_manual(
            breaks = c(
                'knn', 'lr',
                'tree.medium',  'tree.small'
            ),
            labels = c(
                '19-NN (Box-Cox + Simplified)', 'Logistic Regression',
                'Decision Tree (minsplit = 30)',  'Decision Tree (minsplit = 45)'
            ),
            values=c(
                '#ee3532', '#4a2574',
                '#007d80',  '#0050bf'
            )
        ) +
        scale_fill_manual(
            breaks = c(
                'knn', 'lr',
                'tree.medium',  'tree.small'
            ),
            labels = c(
                '19-NN (Box-Cox + Simplified)', 'Logistic Regression',
                'Decision Tree (minsplit = 30)',  'Decision Tree (minsplit = 45)'
            ),
            values=c(
                '#ee3532', '#4a2574',
                '#007d80',  '#0050bf'
            )
        ) +
        theme(
            # legend.position = 'inside',
            # legend.direction = 'horizontal',
            legend.box = 'horizontal',
            legend.box.just = 'right',
            legend.justification = c('right', 'bottom'),
            legend.background = element_blank(),
            strip.background = element_blank(),
            strip.text.x = element_blank()
        )
) %>%
        .save_and_display(
            'ROC curve and AUC for each model',
            width = 960, height = 640,
            '../figures/40.ROCs.pdf'
        )
```




```
final.TPR.df <- final.preds %>% .final.perfs('tpr', 'cutoff')
```

```
## Joining with `by = join_by(tpr, cutoff, Name)`
## Joining with `by = join_by(tpr, cutoff, Name)`
## Joining with `by = join_by(tpr, cutoff, Name)`
```

```
final.TPR.df
```

```
##            tpr      cutoff        Name
## 1   0.00000000         Inf         knn
## 2   0.19254658 1.000000000         knn
## 3   0.36645963 0.947368421         knn
## 4   0.54037267 0.894736842         knn
## 5   0.55900621 0.850000000         knn
## 6   0.60248447 0.842105263         knn
## 7   0.65838509 0.789473684         knn
## 8   0.72049689 0.736842105         knn
## 9   0.72670807 0.700000000         knn
## 10  0.81366460 0.684210526         knn
## 11  0.81987578 0.650000000         knn
## 12  0.86956522 0.631578947         knn
## 13  0.87577640 0.600000000         knn
## 14  0.89440994 0.578947368         knn
## 15  0.91304348 0.526315789         knn
## 16  0.95031056 0.473684211         knn
## 17  0.97515528 0.421052632         knn
## 18  0.97515528 0.400000000         knn
## 19  0.97515528 0.368421053         knn
```

```
## 20  0.98136646 0.315789474         knn
## 21  0.98136646 0.263157895         knn
## 22  0.98136646 0.210526316         knn
## 23  0.98136646 0.157894737         knn
## 24  0.99378882 0.105263158         knn
## 25  1.00000000 0.052631579         knn
## 26  1.00000000 0.050000000         knn
## 27  1.00000000 0.000000000         knn
## 28  0.00000000         Inf          lr
## 29  0.00621118 0.997189666          lr
## 30  0.01242236 0.995799046          lr
## 31  0.01863354 0.994840282          lr
## 32  0.02484472 0.994336395          lr
## 33  0.03105590 0.992647783          lr
## 34  0.03726708 0.991328023          lr
## 35  0.04347826 0.990635590          lr
## 36  0.04968944 0.989617533          lr
## 37  0.05590062 0.988278850          lr
## 38  0.06211180 0.987448189          lr
## 39  0.06832298 0.985789322          lr
## 40  0.07453416 0.983820979          lr
## 41  0.08074534 0.983554313          lr
## 42  0.08695652 0.982869881          lr
## 43  0.09316770 0.982812048          lr
## 44  0.09937888 0.982797743          lr
## 45  0.10559006 0.981009981          lr
## 46  0.11180124 0.980332335          lr
## 47  0.11801242 0.980307120          lr
## 48  0.12422360 0.979649219          lr
## 49  0.13043478 0.978040720          lr
## 50  0.13664596 0.977851279          lr
## 51  0.14285714 0.976672468          lr
## 52  0.14906832 0.976245956          lr
## 53  0.15527950 0.976074037          lr
## 54  0.16149068 0.975066278          lr
## 55  0.16770186 0.974752156          lr
## 56  0.17391304 0.974661114          lr
## 57  0.18012422 0.974391498          lr
## 58  0.18633540 0.974305087          lr
## 59  0.19254658 0.974154107          lr
## 60  0.19875776 0.973472906          lr
## 61  0.20496894 0.971254389          lr
## 62  0.20496894 0.971067492          lr
## 63  0.21118012 0.969292770          lr
## 64  0.21739130 0.968221520          lr
## 65  0.22360248 0.968202094          lr
## 66  0.22981366 0.966124473          lr
## 67  0.23602484 0.965608338          lr
## 68  0.24223602 0.963981701          lr
## 69  0.24844720 0.962323728          lr
## 70  0.25465839 0.956662711          lr
## 71  0.26086957 0.956433480          lr
## 72  0.26708075 0.956315302          lr
## 73  0.27329193 0.956286816          lr
```

```
## 74  0.27950311 0.953456606          lr
## 75  0.28571429 0.950564032          lr
## 76  0.29192547 0.950285333          lr
## 77  0.29813665 0.946606440          lr
## 78  0.30434783 0.944904968          lr
## 79  0.31055901 0.944035183          lr
## 80  0.31677019 0.943699518          lr
## 81  0.32298137 0.941853774          lr
## 82  0.32919255 0.941788922          lr
## 83  0.32919255 0.941696173          lr
## 84  0.33540373 0.941269161          lr
## 85  0.34161491 0.939400944          lr
## 86  0.34782609 0.938337826          lr
## 87  0.35403727 0.937511693          lr
## 88  0.35403727 0.935876399          lr
## 89  0.36024845 0.935853283          lr
## 90  0.36645963 0.934628958          lr
## 91  0.37267081 0.932331746          lr
## 92  0.37888199 0.928804008          lr
## 93  0.38509317 0.927183801          lr
## 94  0.39130435 0.925966920          lr
## 95  0.39751553 0.924485102          lr
## 96  0.40372671 0.923291028          lr
## 97  0.40993789 0.915992420          lr
## 98  0.41614907 0.914235207          lr
## 99  0.42236025 0.914219986          lr
## 100 0.42857143 0.913781719          lr
## 101 0.43478261 0.913695336          lr
## 102 0.44099379 0.911420709          lr
## 103 0.44720497 0.909824591          lr
## 104 0.45341615 0.906774842          lr
## 105 0.45962733 0.904542693          lr
## 106 0.46583851 0.902446000          lr
## 107 0.47204969 0.897465923          lr
## 108 0.47826087 0.892947757          lr
## 109 0.48447205 0.885818199          lr
## 110 0.49068323 0.883131445          lr
## 111 0.49689441 0.876633969          lr
## 112 0.49689441 0.869345886          lr
## 113 0.50310559 0.869267307          lr
## 114 0.50931677 0.866531387          lr
## 115 0.51552795 0.866324228          lr
## 116 0.52173913 0.865581487          lr
## 117 0.52795031 0.865002606          lr
## 118 0.53416149 0.862435143          lr
## 119 0.54037267 0.862199811          lr
## 120 0.54658385 0.861649858          lr
## 121 0.55279503 0.860460270          lr
## 122 0.55900621 0.860317890          lr
## 123 0.56521739 0.859871253          lr
## 124 0.57142857 0.855485009          lr
## 125 0.57142857 0.852786104          lr
## 126 0.57763975 0.850264140          lr
## 127 0.58385093 0.841640499          lr
```

```
## 128 0.59006211 0.840731192          lr
## 129 0.59627329 0.839528754          lr
## 130 0.60248447 0.836689461          lr
## 131 0.60869565 0.827240589          lr
## 132 0.61490683 0.826825480          lr
## 133 0.62111801 0.819563022          lr
## 134 0.62111801 0.816898302          lr
## 135 0.62732919 0.812455662          lr
## 136 0.63354037 0.804955552          lr
## 137 0.63975155 0.804172370          lr
## 138 0.64596273 0.798607903          lr
## 139 0.65217391 0.795032730          lr
## 140 0.65838509 0.793612101          lr
## 141 0.66459627 0.793440657          lr
## 142 0.67080745 0.787421427          lr
## 143 0.67701863 0.783515020          lr
## 144 0.67701863 0.778547065          lr
## 145 0.67701863 0.776205221          lr
## 146 0.68322981 0.773946202          lr
## 147 0.68944099 0.772347639          lr
## 148 0.68944099 0.765449229          lr
## 149 0.69565217 0.759104130          lr
## 150 0.70186335 0.755808751          lr
## 151 0.70807453 0.751527175          lr
## 152 0.71428571 0.748850485          lr
## 153 0.71428571 0.745836336          lr
## 154 0.72049689 0.744866546          lr
## 155 0.72049689 0.743848158          lr
## 156 0.72670807 0.742663920          lr
## 157 0.73291925 0.738589968          lr
## 158 0.73913043 0.737128335          lr
## 159 0.73913043 0.729413184          lr
## 160 0.74534161 0.720409344          lr
## 161 0.75155280 0.720057890          lr
## 162 0.75776398 0.711826152          lr
## 163 0.75776398 0.709171360          lr
## 164 0.75776398 0.707375755          lr
## 165 0.75776398 0.703676733          lr
## 166 0.76397516 0.700058071          lr
## 167 0.77018634 0.697272662          lr
## 168 0.77639752 0.696960381          lr
## 169 0.78260870 0.694135985          lr
## 170 0.78260870 0.690400261          lr
## 171 0.78881988 0.689668806          lr
## 172 0.78881988 0.689399548          lr
## 173 0.80124224 0.681164555          lr
## 174 0.80745342 0.677646259          lr
## 175 0.81366460 0.676964360          lr
## 176 0.81366460 0.676499539          lr
## 177 0.81366460 0.667512882          lr
## 178 0.81366460 0.664451823          lr
## 179 0.81987578 0.658547509          lr
## 180 0.82608696 0.658222843          lr
## 181 0.82608696 0.653954985          lr
```

```
## 182 0.83229814 0.651761422          lr
## 183 0.83850932 0.645218317          lr
## 184 0.84472050 0.626823010          lr
## 185 0.85093168 0.625426364          lr
## 186 0.85714286 0.624382695          lr
## 187 0.85714286 0.617906029          lr
## 188 0.86335404 0.607359728          lr
## 189 0.86956522 0.604761784          lr
## 190 0.87577640 0.599980294          lr
## 191 0.88198758 0.583883788          lr
## 192 0.88819876 0.573992489          lr
## 193 0.88819876 0.565075394          lr
## 194 0.88819876 0.563661253          lr
## 195 0.89440994 0.560211647          lr
## 196 0.89440994 0.558900868          lr
## 197 0.90062112 0.556534049          lr
## 198 0.90062112 0.548495622          lr
## 199 0.90683230 0.537662996          lr
## 200 0.90683230 0.524329456          lr
## 201 0.91304348 0.520419452          lr
## 202 0.91304348 0.510945075          lr
## 203 0.91304348 0.490491801          lr
## 204 0.91304348 0.490051585          lr
## 205 0.91304348 0.471868597          lr
## 206 0.91304348 0.455707546          lr
## 207 0.91925466 0.441678876          lr
## 208 0.91925466 0.435668784          lr
## 209 0.92546584 0.432930825          lr
## 210 0.93167702 0.421344443          lr
## 211 0.93167702 0.401383965          lr
## 212 0.93167702 0.396954971          lr
## 213 0.93788820 0.389753218          lr
## 214 0.93788820 0.353440463          lr
## 215 0.93788820 0.351981422          lr
## 216 0.94409938 0.341446810          lr
## 217 0.94409938 0.325076068          lr
## 218 0.94409938 0.298353913          lr
## 219 0.94409938 0.298160373          lr
## 220 0.94409938 0.295384464          lr
## 221 0.94409938 0.289005473          lr
## 222 0.94409938 0.288379504          lr
## 223 0.94409938 0.270500654          lr
## 224 0.94409938 0.268615387          lr
## 225 0.94409938 0.266825730          lr
## 226 0.94409938 0.258960819          lr
## 227 0.95031056 0.243441676          lr
## 228 0.95031056 0.206353199          lr
## 229 0.95031056 0.202450330          lr
## 230 0.95031056 0.197632294          lr
## 231 0.95652174 0.195228648          lr
## 232 0.95652174 0.193698146          lr
## 233 0.95652174 0.176244246          lr
## 234 0.95652174 0.174040733          lr
## 235 0.95652174 0.170749659          lr
```

```
## 236 0.95652174 0.165555839          lr
## 237 0.95652174 0.163533816          lr
## 238 0.96273292 0.155027097          lr
## 239 0.96273292 0.153344734          lr
## 240 0.96273292 0.151476151          lr
## 241 0.96894410 0.141114553          lr
## 242 0.96894410 0.140657653          lr
## 243 0.96894410 0.136254888          lr
## 244 0.96894410 0.134329255          lr
## 245 0.96894410 0.129919912          lr
## 246 0.96894410 0.116874731          lr
## 247 0.96894410 0.114802400          lr
## 248 0.97515528 0.109548688          lr
## 249 0.97515528 0.107919528          lr
## 250 0.97515528 0.106625962          lr
## 251 0.97515528 0.104428072          lr
## 252 0.97515528 0.103380638          lr
## 253 0.97515528 0.093793590          lr
## 254 0.98136646 0.081822065          lr
## 255 0.98136646 0.079409913          lr
## 256 0.98136646 0.074197994          lr
## 257 0.98757764 0.072399285          lr
## 258 0.98757764 0.067199602          lr
## 259 0.98757764 0.067067746          lr
## 260 0.99378882 0.064102839          lr
## 261 0.99378882 0.062790322          lr
## 262 0.99378882 0.059250629          lr
## 263 0.99378882 0.058716626          lr
## 264 0.99378882 0.057660237          lr
## 265 0.99378882 0.057559349          lr
## 266 0.99378882 0.057469338          lr
## 267 0.99378882 0.057426842          lr
## 268 0.99378882 0.053798276          lr
## 269 0.99378882 0.053204163          lr
## 270 0.99378882 0.052778896          lr
## 271 0.99378882 0.050483747          lr
## 272 0.99378882 0.047896904          lr
## 273 0.99378882 0.047515705          lr
## 274 1.00000000 0.047180497          lr
## 275 1.00000000 0.045085649          lr
## 276 1.00000000 0.043480821          lr
## 277 1.00000000 0.043318597          lr
## 278 1.00000000 0.043254732          lr
## 279 1.00000000 0.041238110          lr
## 280 1.00000000 0.039635112          lr
## 281 1.00000000 0.038796090          lr
## 282 1.00000000 0.038705196          lr
## 283 1.00000000 0.036561337          lr
## 284 1.00000000 0.033565705          lr
## 285 1.00000000 0.033155446          lr
## 286 1.00000000 0.033012697          lr
## 287 1.00000000 0.031043223          lr
## 288 1.00000000 0.030435899          lr
## 289 1.00000000 0.029305262          lr
```

```
## 290 1.00000000 0.026671880          lr
## 291 1.00000000 0.025325930          lr
## 292 1.00000000 0.023617286          lr
## 293 1.00000000 0.023315398          lr
## 294 1.00000000 0.021257680          lr
## 295 1.00000000 0.021116657          lr
## 296 1.00000000 0.021016552          lr
## 297 1.00000000 0.019693522          lr
## 298 1.00000000 0.017843412          lr
## 299 1.00000000 0.016792231          lr
## 300 1.00000000 0.013302789          lr
## 301 1.00000000 0.011797820          lr
## 302 1.00000000 0.011089713          lr
## 303 1.00000000 0.011045078          lr
## 304 1.00000000 0.010434003          lr
## 305 1.00000000 0.008950515          lr
## 306 1.00000000 0.008535923          lr
## 307 1.00000000 0.008255904          lr
## 308 1.00000000 0.007837263          lr
## 309 1.00000000 0.007528373          lr
## 310 1.00000000 0.006921597          lr
## 311 1.00000000 0.006553938          lr
## 312 1.00000000 0.006536330          lr
## 313 1.00000000 0.006284399          lr
## 314 1.00000000 0.006190114          lr
## 315 1.00000000 0.006073408          lr
## 316 1.00000000 0.005110959          lr
## 317 1.00000000 0.004863847          lr
## 318 1.00000000 0.004097913          lr
## 319 1.00000000 0.004057368          lr
## 320 1.00000000 0.003791678          lr
## 321 1.00000000 0.003580751          lr
## 322 1.00000000 0.003284462          lr
## 323 1.00000000 0.002904543          lr
## 324 1.00000000 0.002830813          lr
## 325 1.00000000 0.001024204          lr
## 326 0.00000000         Inf  tree.small
## 327 0.62111801 0.892857143  tree.small
## 328 0.77639752 0.781250000  tree.small
## 329 0.91304348 0.666666667  tree.small
## 330 0.92546584 0.166666667  tree.small
## 331 0.98757764 0.111111111  tree.small
## 332 1.00000000 0.105263158  tree.small
## 333 0.00000000         Inf tree.medium
## 334 0.62111801 0.892857143 tree.medium
## 335 0.77639752 0.781250000 tree.medium
## 336 0.89440994 0.703703704 tree.medium
## 337 0.92546584 0.277777778 tree.medium
## 338 0.98757764 0.111111111 tree.medium
## 339 1.00000000 0.105263158 tree.medium
```

```
final.precision.df <- final.preds %>% .final.perfs('prec', 'cutoff')
```

```
## Joining with `by = join_by(prec, cutoff, Name)`
## Joining with `by = join_by(prec, cutoff, Name)`
```

```
## Joining with `by = join_by(prec, cutoff, Name)`
```

```
final.precision.df
```

```
##          prec      cutoff        Name
## 1         NaN         Inf         knn
## 2   1.0000000 1.000000000         knn
## 3   1.0000000 0.947368421         knn
## 4   0.9560440 0.894736842         knn
## 5   0.9574468 0.850000000         knn
## 6   0.9326923 0.842105263         knn
## 7   0.9137931 0.789473684         knn
## 8   0.8854962 0.736842105         knn
## 9   0.8863636 0.700000000         knn
## 10  0.8791946 0.684210526         knn
## 11  0.8800000 0.650000000         knn
## 12  0.8484848 0.631578947         knn
## 13  0.8493976 0.600000000         knn
## 14  0.8228571 0.578947368         knn
## 15  0.8121547 0.526315789         knn
## 16  0.7806122 0.473684211         knn
## 17  0.7772277 0.421052632         knn
## 18  0.7733990 0.400000000         knn
## 19  0.7511962 0.368421053         knn
## 20  0.7314815 0.315789474         knn
## 21  0.7181818 0.263157895         knn
## 22  0.7117117 0.210526316         knn
## 23  0.6839827 0.157894737         knn
## 24  0.6584362 0.105263158         knn
## 25  0.5897436 0.052631579         knn
## 26  0.5875912 0.050000000         knn
## 27  0.5402685 0.000000000         knn
## 28        NaN         Inf          lr
## 29  1.0000000 0.997189666          lr
## 30  1.0000000 0.995799046          lr
## 31  1.0000000 0.994840282          lr
## 32  1.0000000 0.994336395          lr
## 33  1.0000000 0.992647783          lr
## 34  1.0000000 0.991328023          lr
## 35  1.0000000 0.990635590          lr
## 36  1.0000000 0.989617533          lr
## 37  1.0000000 0.988278850          lr
## 38  1.0000000 0.987448189          lr
## 39  1.0000000 0.985789322          lr
## 40  1.0000000 0.983820979          lr
## 41  1.0000000 0.983554313          lr
## 42  1.0000000 0.982869881          lr
## 43  1.0000000 0.982812048          lr
## 44  1.0000000 0.982797743          lr
## 45  1.0000000 0.981009981          lr
## 46  1.0000000 0.980332335          lr
## 47  1.0000000 0.980307120          lr
## 48  1.0000000 0.979649219          lr
## 49  1.0000000 0.978040720          lr
## 50  1.0000000 0.977851279          lr
```

```
## 51  1.0000000 0.976672468          lr
## 52  1.0000000 0.976245956          lr
## 53  1.0000000 0.976074037          lr
## 54  1.0000000 0.975066278          lr
## 55  1.0000000 0.974752156          lr
## 56  1.0000000 0.974661114          lr
## 57  1.0000000 0.974391498          lr
## 58  1.0000000 0.974305087          lr
## 59  1.0000000 0.974154107          lr
## 60  1.0000000 0.973472906          lr
## 61  1.0000000 0.971254389          lr
## 62  0.9705882 0.971067492          lr
## 63  0.9714286 0.969292770          lr
## 64  0.9722222 0.968221520          lr
## 65  0.9729730 0.968202094          lr
## 66  0.9736842 0.966124473          lr
## 67  0.9743590 0.965608338          lr
## 68  0.9750000 0.963981701          lr
## 69  0.9756098 0.962323728          lr
## 70  0.9761905 0.956662711          lr
## 71  0.9767442 0.956433480          lr
## 72  0.9772727 0.956315302          lr
## 73  0.9777778 0.956286816          lr
## 74  0.9782609 0.953456606          lr
## 75  0.9787234 0.950564032          lr
## 76  0.9791667 0.950285333          lr
## 77  0.9795918 0.946606440          lr
## 78  0.9800000 0.944904968          lr
## 79  0.9803922 0.944035183          lr
## 80  0.9807692 0.943699518          lr
## 81  0.9811321 0.941853774          lr
## 82  0.9814815 0.941788922          lr
## 83  0.9636364 0.941696173          lr
## 84  0.9642857 0.941269161          lr
## 85  0.9649123 0.939400944          lr
## 86  0.9655172 0.938337826          lr
## 87  0.9661017 0.937511693          lr
## 88  0.9500000 0.935876399          lr
## 89  0.9508197 0.935853283          lr
## 90  0.9516129 0.934628958          lr
## 91  0.9523810 0.932331746          lr
## 92  0.9531250 0.928804008          lr
## 93  0.9538462 0.927183801          lr
## 94  0.9545455 0.925966920          lr
## 95  0.9552239 0.924485102          lr
## 96  0.9558824 0.923291028          lr
## 97  0.9565217 0.915992420          lr
## 98  0.9571429 0.914235207          lr
## 99  0.9577465 0.914219986          lr
## 100 0.9583333 0.913781719          lr
## 101 0.9589041 0.913695336          lr
## 102 0.9594595 0.911420709          lr
## 103 0.9600000 0.909824591          lr
## 104 0.9605263 0.906774842          lr
```

```
## 105 0.9610390 0.904542693          lr
## 106 0.9615385 0.902446000          lr
## 107 0.9620253 0.897465923          lr
## 108 0.9625000 0.892947757          lr
## 109 0.9629630 0.885818199          lr
## 110 0.9634146 0.883131445          lr
## 111 0.9638554 0.876633969          lr
## 112 0.9523810 0.869345886          lr
## 113 0.9529412 0.869267307          lr
## 114 0.9534884 0.866531387          lr
## 115 0.9540230 0.866324228          lr
## 116 0.9545455 0.865581487          lr
## 117 0.9550562 0.865002606          lr
## 118 0.9555556 0.862435143          lr
## 119 0.9560440 0.862199811          lr
## 120 0.9565217 0.861649858          lr
## 121 0.9569892 0.860460270          lr
## 122 0.9574468 0.860317890          lr
## 123 0.9578947 0.859871253          lr
## 124 0.9583333 0.855485009          lr
## 125 0.9484536 0.852786104          lr
## 126 0.9489796 0.850264140          lr
## 127 0.9494949 0.841640499          lr
## 128 0.9500000 0.840731192          lr
## 129 0.9504950 0.839528754          lr
## 130 0.9509804 0.836689461          lr
## 131 0.9514563 0.827240589          lr
## 132 0.9519231 0.826825480          lr
## 133 0.9523810 0.819563022          lr
## 134 0.9433962 0.816898302          lr
## 135 0.9439252 0.812455662          lr
## 136 0.9444444 0.804955552          lr
## 137 0.9449541 0.804172370          lr
## 138 0.9454545 0.798607903          lr
## 139 0.9459459 0.795032730          lr
## 140 0.9464286 0.793612101          lr
## 141 0.9469027 0.793440657          lr
## 142 0.9473684 0.787421427          lr
## 143 0.9478261 0.783515020          lr
## 144 0.9396552 0.778547065          lr
## 145 0.9316239 0.776205221          lr
## 146 0.9322034 0.773946202          lr
## 147 0.9327731 0.772347639          lr
## 148 0.9250000 0.765449229          lr
## 149 0.9256198 0.759104130          lr
## 150 0.9262295 0.755808751          lr
## 151 0.9268293 0.751527175          lr
## 152 0.9274194 0.748850485          lr
## 153 0.9200000 0.745836336          lr
## 154 0.9206349 0.744866546          lr
## 155 0.9133858 0.743848158          lr
## 156 0.9140625 0.742663920          lr
## 157 0.9147287 0.738589968          lr
## 158 0.9153846 0.737128335          lr
```

```
## 159 0.9083969 0.729413184          lr
## 160 0.9090909 0.720409344          lr
## 161 0.9097744 0.720057890          lr
## 162 0.9104478 0.711826152          lr
## 163 0.9037037 0.709171360          lr
## 164 0.8970588 0.707375755          lr
## 165 0.8905109 0.703676733          lr
## 166 0.8913043 0.700058071          lr
## 167 0.8920863 0.697272662          lr
## 168 0.8928571 0.696960381          lr
## 169 0.8936170 0.694135985          lr
## 170 0.8873239 0.690400261          lr
## 171 0.8881119 0.689668806          lr
## 172 0.8819444 0.689399548          lr
## 173 0.8835616 0.681164555          lr
## 174 0.8843537 0.677646259          lr
## 175 0.8851351 0.676964360          lr
## 176 0.8791946 0.676499539          lr
## 177 0.8733333 0.667512882          lr
## 178 0.8675497 0.664451823          lr
## 179 0.8684211 0.658547509          lr
## 180 0.8692810 0.658222843          lr
## 181 0.8636364 0.653954985          lr
## 182 0.8645161 0.651761422          lr
## 183 0.8653846 0.645218317          lr
## 184 0.8662420 0.626823010          lr
## 185 0.8670886 0.625426364          lr
## 186 0.8679245 0.624382695          lr
## 187 0.8625000 0.617906029          lr
## 188 0.8633540 0.607359728          lr
## 189 0.8641975 0.604761784          lr
## 190 0.8650307 0.599980294          lr
## 191 0.8658537 0.583883788          lr
## 192 0.8666667 0.573992489          lr
## 193 0.8614458 0.565075394          lr
## 194 0.8562874 0.563661253          lr
## 195 0.8571429 0.560211647          lr
## 196 0.8520710 0.558900868          lr
## 197 0.8529412 0.556534049          lr
## 198 0.8479532 0.548495622          lr
## 199 0.8488372 0.537662996          lr
## 200 0.8439306 0.524329456          lr
## 201 0.8448276 0.520419452          lr
## 202 0.8400000 0.510945075          lr
## 203 0.8352273 0.490491801          lr
## 204 0.8305085 0.490051585          lr
## 205 0.8258427 0.471868597          lr
## 206 0.8212291 0.455707546          lr
## 207 0.8222222 0.441678876          lr
## 208 0.8176796 0.435668784          lr
## 209 0.8186813 0.432930825          lr
## 210 0.8196721 0.421344443          lr
## 211 0.8152174 0.401383965          lr
## 212 0.8108108 0.396954971          lr
```

```
## 213 0.8118280 0.389753218          lr
## 214 0.8074866 0.353440463          lr
## 215 0.8031915 0.351981422          lr
## 216 0.8042328 0.341446810          lr
## 217 0.8000000 0.325076068          lr
## 218 0.7958115 0.298353913          lr
## 219 0.7916667 0.298160373          lr
## 220 0.7875648 0.295384464          lr
## 221 0.7835052 0.289005473          lr
## 222 0.7794872 0.288379504          lr
## 223 0.7755102 0.270500654          lr
## 224 0.7715736 0.268615387          lr
## 225 0.7676768 0.266825730          lr
## 226 0.7638191 0.258960819          lr
## 227 0.7650000 0.243441676          lr
## 228 0.7611940 0.206353199          lr
## 229 0.7574257 0.202450330          lr
## 230 0.7536946 0.197632294          lr
## 231 0.7549020 0.195228648          lr
## 232 0.7512195 0.193698146          lr
## 233 0.7475728 0.176244246          lr
## 234 0.7439614 0.174040733          lr
## 235 0.7403846 0.170749659          lr
## 236 0.7368421 0.165555839          lr
## 237 0.7333333 0.163533816          lr
## 238 0.7345972 0.155027097          lr
## 239 0.7311321 0.153344734          lr
## 240 0.7276995 0.151476151          lr
## 241 0.7289720 0.141114553          lr
## 242 0.7255814 0.140657653          lr
## 243 0.7222222 0.136254888          lr
## 244 0.7188940 0.134329255          lr
## 245 0.7155963 0.129919912          lr
## 246 0.7123288 0.116874731          lr
## 247 0.7090909 0.114802400          lr
## 248 0.7104072 0.109548688          lr
## 249 0.7072072 0.107919528          lr
## 250 0.7040359 0.106625962          lr
## 251 0.7008929 0.104428072          lr
## 252 0.6977778 0.103380638          lr
## 253 0.6946903 0.093793590          lr
## 254 0.6960352 0.081822065          lr
## 255 0.6929825 0.079409913          lr
## 256 0.6899563 0.074197994          lr
## 257 0.6913043 0.072399285          lr
## 258 0.6883117 0.067199602          lr
## 259 0.6853448 0.067067746          lr
## 260 0.6866953 0.064102839          lr
## 261 0.6837607 0.062790322          lr
## 262 0.6808511 0.059250629          lr
## 263 0.6779661 0.058716626          lr
## 264 0.6751055 0.057660237          lr
## 265 0.6722689 0.057559349          lr
## 266 0.6694561 0.057469338          lr
```

```
## 267 0.6666667 0.057426842         lr
## 268 0.6639004 0.053798276         lr
## 269 0.6611570 0.053204163         lr
## 270 0.6584362 0.052778896         lr
## 271 0.6557377 0.050483747         lr
## 272 0.6530612 0.047896904         lr
## 273 0.6504065 0.047515705         lr
## 274 0.6518219 0.047180497         lr
## 275 0.6491935 0.045085649         lr
## 276 0.6465863 0.043480821         lr
## 277 0.6440000 0.043318597         lr
## 278 0.6414343 0.043254732         lr
## 279 0.6388889 0.041238110         lr
## 280 0.6363636 0.039635112         lr
## 281 0.6338583 0.038796090         lr
## 282 0.6313725 0.038705196         lr
## 283 0.6289062 0.036561337         lr
## 284 0.6264591 0.033565705         lr
## 285 0.6240310 0.033155446         lr
## 286 0.6216216 0.033012697         lr
## 287 0.6192308 0.031043223         lr
## 288 0.6168582 0.030435899         lr
## 289 0.6145038 0.029305262         lr
## 290 0.6121673 0.026671880         lr
## 291 0.6098485 0.025325930         lr
## 292 0.6075472 0.023617286         lr
## 293 0.6052632 0.023315398         lr
## 294 0.6029963 0.021257680         lr
## 295 0.6007463 0.021116657         lr
## 296 0.5985130 0.021016552         lr
## 297 0.5962963 0.019693522         lr
## 298 0.5940959 0.017843412         lr
## 299 0.5919118 0.016792231         lr
## 300 0.5897436 0.013302789         lr
## 301 0.5875912 0.011797820         lr
## 302 0.5854545 0.011089713         lr
## 303 0.5833333 0.011045078         lr
## 304 0.5812274 0.010434003         lr
## 305 0.5791367 0.008950515         lr
## 306 0.5770609 0.008535923         lr
## 307 0.5750000 0.008255904         lr
## 308 0.5729537 0.007837263         lr
## 309 0.5709220 0.007528373         lr
## 310 0.5689046 0.006921597         lr
## 311 0.5669014 0.006553938         lr
## 312 0.5649123 0.006536330         lr
## 313 0.5629371 0.006284399         lr
## 314 0.5609756 0.006190114         lr
## 315 0.5590278 0.006073408         lr
## 316 0.5570934 0.005110959         lr
## 317 0.5551724 0.004863847         lr
## 318 0.5532646 0.004097913         lr
## 319 0.5513699 0.004057368         lr
## 320 0.5494881 0.003791678         lr
```

```
## 321 0.5476190 0.003580751          lr
## 322 0.5457627 0.003284462          lr
## 323 0.5439189 0.002904543          lr
## 324 0.5420875 0.002830813          lr
## 325 0.5402685 0.001024204          lr
## 326       NaN         Inf  tree.small
## 327 0.8928571 0.892857143  tree.small
## 328 0.8680556 0.781250000  tree.small
## 329 0.8305085 0.666666667  tree.small
## 330 0.7883598 0.166666667  tree.small
## 331 0.5698925 0.111111111  tree.small
## 332 0.5402685 0.105263158  tree.small
## 333       NaN         Inf tree.medium
## 334 0.8928571 0.892857143 tree.medium
## 335 0.8680556 0.781250000 tree.medium
## 336 0.8421053 0.703703704 tree.medium
## 337 0.7883598 0.277777778 tree.medium
## 338 0.5698925 0.111111111 tree.medium
## 339 0.5402685 0.105263158 tree.medium
```

```
final.TPR.precision.df <- inner_join(final.TPR.df, final.precision.df) %>%
    pivot_longer(cols = c('tpr', 'prec'), names_to = 'metric', values_to = 'values')
```

```
## Joining with `by = join_by(cutoff, Name)`
```

```
final.TPR.precision.50 <- final.TPR.precision.df %>%
    filter(0 <= cutoff, cutoff <= 1) %>%
    filter(0 <= values, values <= 1) %>%
    group_by(Name, metric) %>%
    summarise(value_50 = approxfun(cutoff, values, n = 100)(.5))
```

```
## `summarise()` has grouped output by 'Name'. You can override using the
## `.groups` argument.
```

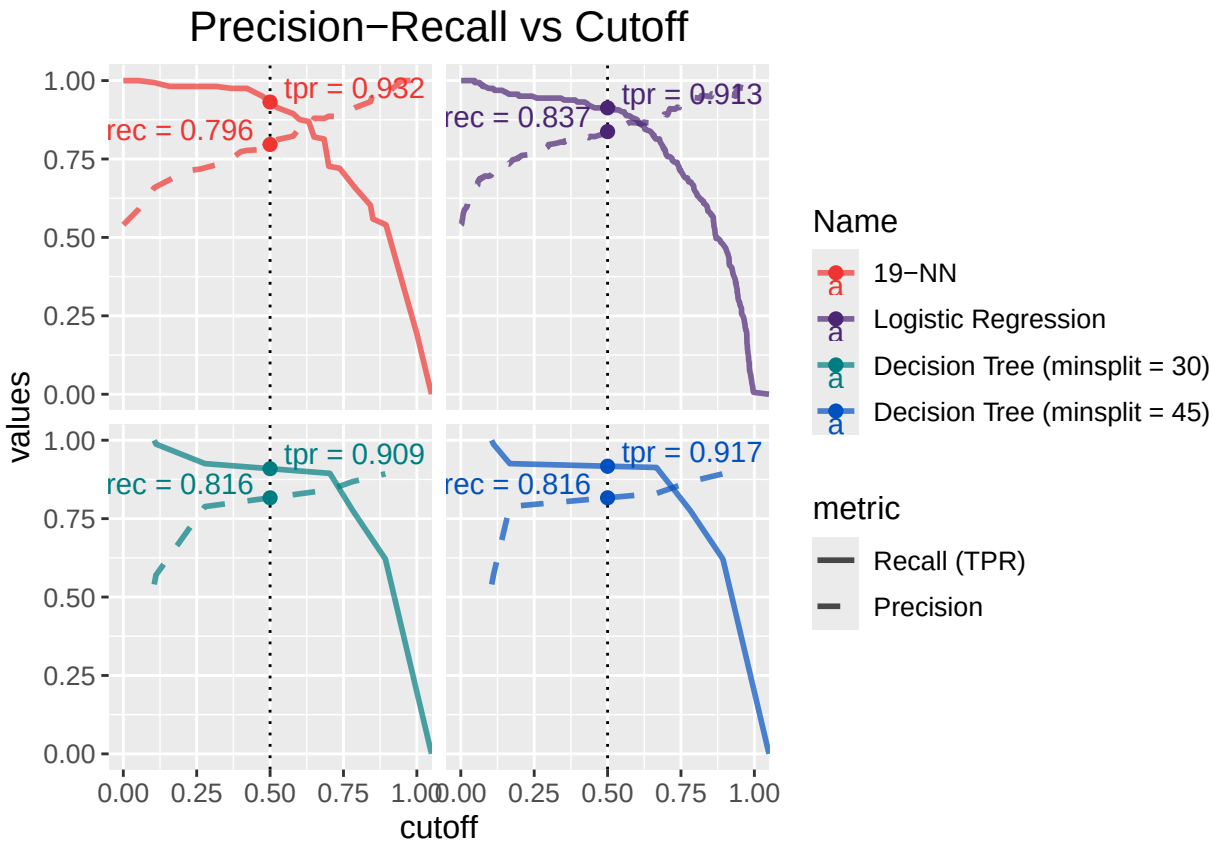
```
final.TPR.precision.75 <- final.TPR.precision.df %>%
    filter(0 <= cutoff, cutoff <= 1) %>%
    filter(0 <= values, values <= 1) %>%
    group_by(Name, metric) %>%
    summarise(value_75 = approxfun(cutoff, values, n = 100)(.75))
```

```
## `summarise()` has grouped output by 'Name'. You can override using the
## `.groups` argument.
```

```
(final.TPR.precision.df %>% ggplot() +
    facet_wrap(~ Name) +
    geom_line(
        mapping = aes(x = cutoff, y = values, col = Name, linetype = metric),
        size = 1, alpha = .7
    ) +
    geom_vline(
        xintercept = .5,
        linetype = 'dotted',
        size = .5
    ) +
    geom_point(
        data = final.TPR.precision.50,
        mapping = aes(x = 0.5, y = value_50, col = Name),
```

```
        size = 2
    ) +
    geom_text(
        data = final.TPR.precision.50,
        mapping = aes(
            x = 0.5, y = value_50, col = Name,
            label = paste0(metric, ' = ', value_50 %>% round(3)),
            hjust = metric %>% sapply(\(metric) if (metric == 'tpr') -0.1 else 1.1)
        ),
        vjust = -0.2
    ) +
        scale_color_manual(
            breaks = c(
                'knn', 'lr',
                'tree.medium',  'tree.small'
            ),
            labels = c(
                '19-NN', 'Logistic Regression',
                'Decision Tree (minsplit = 30)',  'Decision Tree (minsplit = 45)'
            ),
            values=c(
                '#ee3532', '#4a2574',
                '#007d80',  '#0050bf'
            )
        ) +
        scale_linetype_manual(
            breaks = c( 'tpr', 'prec' ),
            labels = c( 'Recall (TPR)', 'Precision' ),
            values=c( 'solid', 'dashed' )
        ) +
        theme(
            # legend.position = 'inside',
            # legend.direction = 'horizontal',
            # legend.box = 'horizontal',
            # legend.box.just = 'right',
            # legend.justification = c('right', 'bottom'),
            legend.background = element_blank(),
            strip.background = element_blank(),
            strip.text.x = element_blank()
        )
) %>%
        .save_and_display(
            'Precision-Recall vs Cutoff',
            width = 960, height = 640,
            '../figures/40.TPRs.pdf'
        )
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```
final.prec.recall.df <- final.preds %>% .final.perfs('prec', 'tpr')
```

```
## Joining with `by = join_by(prec, tpr, Name)`
## Joining with `by = join_by(prec, tpr, Name)`
## Joining with `by = join_by(prec, tpr, Name)`
```

```
final.prec.recall.df
```

```
##          prec        tpr        Name
## 1         NaN 0.00000000         knn
## 2   1.0000000 0.19254658         knn
## 3   1.0000000 0.36645963         knn
## 4   0.9560440 0.54037267         knn
## 5   0.9574468 0.55900621         knn
## 6   0.9326923 0.60248447         knn
## 7   0.9137931 0.65838509         knn
## 8   0.8854962 0.72049689         knn
## 9   0.8863636 0.72670807         knn
## 10  0.8791946 0.81366460         knn
## 11  0.8800000 0.81987578         knn
## 12  0.8484848 0.86956522         knn
## 13  0.8493976 0.87577640         knn
## 14  0.8228571 0.89440994         knn
## 15  0.8121547 0.91304348         knn
## 16  0.7806122 0.95031056         knn
## 17  0.7772277 0.97515528         knn
## 18  0.7733990 0.97515528         knn
## 19  0.7511962 0.97515528         knn
```

```
## 20  0.7314815 0.98136646         knn
## 21  0.7181818 0.98136646         knn
## 22  0.7117117 0.98136646         knn
## 23  0.6839827 0.98136646         knn
## 24  0.6584362 0.99378882         knn
## 25  0.5897436 1.00000000         knn
## 26  0.5875912 1.00000000         knn
## 27  0.5402685 1.00000000         knn
## 28        NaN 0.00000000          lr
## 29  1.0000000 0.00621118          lr
## 30  1.0000000 0.01242236          lr
## 31  1.0000000 0.01863354          lr
## 32  1.0000000 0.02484472          lr
## 33  1.0000000 0.03105590          lr
## 34  1.0000000 0.03726708          lr
## 35  1.0000000 0.04347826          lr
## 36  1.0000000 0.04968944          lr
## 37  1.0000000 0.05590062          lr
## 38  1.0000000 0.06211180          lr
## 39  1.0000000 0.06832298          lr
## 40  1.0000000 0.07453416          lr
## 41  1.0000000 0.08074534          lr
## 42  1.0000000 0.08695652          lr
## 43  1.0000000 0.09316770          lr
## 44  1.0000000 0.09937888          lr
## 45  1.0000000 0.10559006          lr
## 46  1.0000000 0.11180124          lr
## 47  1.0000000 0.11801242          lr
## 48  1.0000000 0.12422360          lr
## 49  1.0000000 0.13043478          lr
## 50  1.0000000 0.13664596          lr
## 51  1.0000000 0.14285714          lr
## 52  1.0000000 0.14906832          lr
## 53  1.0000000 0.15527950          lr
## 54  1.0000000 0.16149068          lr
## 55  1.0000000 0.16770186          lr
## 56  1.0000000 0.17391304          lr
## 57  1.0000000 0.18012422          lr
## 58  1.0000000 0.18633540          lr
## 59  1.0000000 0.19254658          lr
## 60  1.0000000 0.19875776          lr
## 61  1.0000000 0.20496894          lr
## 62  0.9705882 0.20496894          lr
## 63  0.9714286 0.21118012          lr
## 64  0.9722222 0.21739130          lr
## 65  0.9729730 0.22360248          lr
## 66  0.9736842 0.22981366          lr
## 67  0.9743590 0.23602484          lr
## 68  0.9750000 0.24223602          lr
## 69  0.9756098 0.24844720          lr
## 70  0.9761905 0.25465839          lr
## 71  0.9767442 0.26086957          lr
## 72  0.9772727 0.26708075          lr
## 73  0.9777778 0.27329193          lr
```

```
## 74  0.9782609 0.27950311          lr
## 75  0.9787234 0.28571429          lr
## 76  0.9791667 0.29192547          lr
## 77  0.9795918 0.29813665          lr
## 78  0.9800000 0.30434783          lr
## 79  0.9803922 0.31055901          lr
## 80  0.9807692 0.31677019          lr
## 81  0.9811321 0.32298137          lr
## 82  0.9814815 0.32919255          lr
## 83  0.9636364 0.32919255          lr
## 84  0.9642857 0.33540373          lr
## 85  0.9649123 0.34161491          lr
## 86  0.9655172 0.34782609          lr
## 87  0.9661017 0.35403727          lr
## 88  0.9500000 0.35403727          lr
## 89  0.9508197 0.36024845          lr
## 90  0.9516129 0.36645963          lr
## 91  0.9523810 0.37267081          lr
## 92  0.9531250 0.37888199          lr
## 93  0.9538462 0.38509317          lr
## 94  0.9545455 0.39130435          lr
## 95  0.9552239 0.39751553          lr
## 96  0.9558824 0.40372671          lr
## 97  0.9565217 0.40993789          lr
## 98  0.9571429 0.41614907          lr
## 99  0.9577465 0.42236025          lr
## 100 0.9583333 0.42857143          lr
## 101 0.9589041 0.43478261          lr
## 102 0.9594595 0.44099379          lr
## 103 0.9600000 0.44720497          lr
## 104 0.9605263 0.45341615          lr
## 105 0.9610390 0.45962733          lr
## 106 0.9615385 0.46583851          lr
## 107 0.9620253 0.47204969          lr
## 108 0.9625000 0.47826087          lr
## 109 0.9629630 0.48447205          lr
## 110 0.9634146 0.49068323          lr
## 111 0.9638554 0.49689441          lr
## 112 0.9523810 0.49689441          lr
## 113 0.9529412 0.50310559          lr
## 114 0.9534884 0.50931677          lr
## 115 0.9540230 0.51552795          lr
## 116 0.9545455 0.52173913          lr
## 117 0.9550562 0.52795031          lr
## 118 0.9555556 0.53416149          lr
## 119 0.9560440 0.54037267          lr
## 120 0.9565217 0.54658385          lr
## 121 0.9569892 0.55279503          lr
## 122 0.9574468 0.55900621          lr
## 123 0.9578947 0.56521739          lr
## 124 0.9583333 0.57142857          lr
## 125 0.9484536 0.57142857          lr
## 126 0.9489796 0.57763975          lr
## 127 0.9494949 0.58385093          lr
```

```
## 128 0.9500000 0.59006211          lr
## 129 0.9504950 0.59627329          lr
## 130 0.9509804 0.60248447          lr
## 131 0.9514563 0.60869565          lr
## 132 0.9519231 0.61490683          lr
## 133 0.9523810 0.62111801          lr
## 134 0.9433962 0.62111801          lr
## 135 0.9439252 0.62732919          lr
## 136 0.9444444 0.63354037          lr
## 137 0.9449541 0.63975155          lr
## 138 0.9454545 0.64596273          lr
## 139 0.9459459 0.65217391          lr
## 140 0.9464286 0.65838509          lr
## 141 0.9469027 0.66459627          lr
## 142 0.9473684 0.67080745          lr
## 143 0.9478261 0.67701863          lr
## 144 0.9396552 0.67701863          lr
## 145 0.9316239 0.67701863          lr
## 146 0.9322034 0.68322981          lr
## 147 0.9327731 0.68944099          lr
## 148 0.9250000 0.68944099          lr
## 149 0.9256198 0.69565217          lr
## 150 0.9262295 0.70186335          lr
## 151 0.9268293 0.70807453          lr
## 152 0.9274194 0.71428571          lr
## 153 0.9200000 0.71428571          lr
## 154 0.9206349 0.72049689          lr
## 155 0.9133858 0.72049689          lr
## 156 0.9140625 0.72670807          lr
## 157 0.9147287 0.73291925          lr
## 158 0.9153846 0.73913043          lr
## 159 0.9083969 0.73913043          lr
## 160 0.9090909 0.74534161          lr
## 161 0.9097744 0.75155280          lr
## 162 0.9104478 0.75776398          lr
## 163 0.9037037 0.75776398          lr
## 164 0.8970588 0.75776398          lr
## 165 0.8905109 0.75776398          lr
## 166 0.8913043 0.76397516          lr
## 167 0.8920863 0.77018634          lr
## 168 0.8928571 0.77639752          lr
## 169 0.8936170 0.78260870          lr
## 170 0.8873239 0.78260870          lr
## 171 0.8881119 0.78881988          lr
## 172 0.8819444 0.78881988          lr
## 173 0.8835616 0.80124224          lr
## 174 0.8843537 0.80745342          lr
## 175 0.8851351 0.81366460          lr
## 176 0.8791946 0.81366460          lr
## 177 0.8733333 0.81366460          lr
## 178 0.8675497 0.81366460          lr
## 179 0.8684211 0.81987578          lr
## 180 0.8692810 0.82608696          lr
## 181 0.8636364 0.82608696          lr
```

```
## 182 0.8645161 0.83229814          lr
## 183 0.8653846 0.83850932          lr
## 184 0.8662420 0.84472050          lr
## 185 0.8670886 0.85093168          lr
## 186 0.8679245 0.85714286          lr
## 187 0.8625000 0.85714286          lr
## 188 0.8633540 0.86335404          lr
## 189 0.8641975 0.86956522          lr
## 190 0.8650307 0.87577640          lr
## 191 0.8658537 0.88198758          lr
## 192 0.8666667 0.88819876          lr
## 193 0.8614458 0.88819876          lr
## 194 0.8562874 0.88819876          lr
## 195 0.8571429 0.89440994          lr
## 196 0.8520710 0.89440994          lr
## 197 0.8529412 0.90062112          lr
## 198 0.8479532 0.90062112          lr
## 199 0.8488372 0.90683230          lr
## 200 0.8439306 0.90683230          lr
## 201 0.8448276 0.91304348          lr
## 202 0.8400000 0.91304348          lr
## 203 0.8352273 0.91304348          lr
## 204 0.8305085 0.91304348          lr
## 205 0.8258427 0.91304348          lr
## 206 0.8212291 0.91304348          lr
## 207 0.8222222 0.91925466          lr
## 208 0.8176796 0.91925466          lr
## 209 0.8186813 0.92546584          lr
## 210 0.8196721 0.93167702          lr
## 211 0.8152174 0.93167702          lr
## 212 0.8108108 0.93167702          lr
## 213 0.8118280 0.93788820          lr
## 214 0.8074866 0.93788820          lr
## 215 0.8031915 0.93788820          lr
## 216 0.8042328 0.94409938          lr
## 217 0.8000000 0.94409938          lr
## 218 0.7958115 0.94409938          lr
## 219 0.7916667 0.94409938          lr
## 220 0.7875648 0.94409938          lr
## 221 0.7835052 0.94409938          lr
## 222 0.7794872 0.94409938          lr
## 223 0.7755102 0.94409938          lr
## 224 0.7715736 0.94409938          lr
## 225 0.7676768 0.94409938          lr
## 226 0.7638191 0.94409938          lr
## 227 0.7650000 0.95031056          lr
## 228 0.7611940 0.95031056          lr
## 229 0.7574257 0.95031056          lr
## 230 0.7536946 0.95031056          lr
## 231 0.7549020 0.95652174          lr
## 232 0.7512195 0.95652174          lr
## 233 0.7475728 0.95652174          lr
## 234 0.7439614 0.95652174          lr
## 235 0.7403846 0.95652174          lr
```

```
## 236 0.7368421 0.95652174          lr
## 237 0.7333333 0.95652174          lr
## 238 0.7345972 0.96273292          lr
## 239 0.7311321 0.96273292          lr
## 240 0.7276995 0.96273292          lr
## 241 0.7289720 0.96894410          lr
## 242 0.7255814 0.96894410          lr
## 243 0.7222222 0.96894410          lr
## 244 0.7188940 0.96894410          lr
## 245 0.7155963 0.96894410          lr
## 246 0.7123288 0.96894410          lr
## 247 0.7090909 0.96894410          lr
## 248 0.7104072 0.97515528          lr
## 249 0.7072072 0.97515528          lr
## 250 0.7040359 0.97515528          lr
## 251 0.7008929 0.97515528          lr
## 252 0.6977778 0.97515528          lr
## 253 0.6946903 0.97515528          lr
## 254 0.6960352 0.98136646          lr
## 255 0.6929825 0.98136646          lr
## 256 0.6899563 0.98136646          lr
## 257 0.6913043 0.98757764          lr
## 258 0.6883117 0.98757764          lr
## 259 0.6853448 0.98757764          lr
## 260 0.6866953 0.99378882          lr
## 261 0.6837607 0.99378882          lr
## 262 0.6808511 0.99378882          lr
## 263 0.6779661 0.99378882          lr
## 264 0.6751055 0.99378882          lr
## 265 0.6722689 0.99378882          lr
## 266 0.6694561 0.99378882          lr
## 267 0.6666667 0.99378882          lr
## 268 0.6639004 0.99378882          lr
## 269 0.6611570 0.99378882          lr
## 270 0.6584362 0.99378882          lr
## 271 0.6557377 0.99378882          lr
## 272 0.6530612 0.99378882          lr
## 273 0.6504065 0.99378882          lr
## 274 0.6518219 1.00000000          lr
## 275 0.6491935 1.00000000          lr
## 276 0.6465863 1.00000000          lr
## 277 0.6440000 1.00000000          lr
## 278 0.6414343 1.00000000          lr
## 279 0.6388889 1.00000000          lr
## 280 0.6363636 1.00000000          lr
## 281 0.6338583 1.00000000          lr
## 282 0.6313725 1.00000000          lr
## 283 0.6289062 1.00000000          lr
## 284 0.6264591 1.00000000          lr
## 285 0.6240310 1.00000000          lr
## 286 0.6216216 1.00000000          lr
## 287 0.6192308 1.00000000          lr
## 288 0.6168582 1.00000000          lr
## 289 0.6145038 1.00000000          lr
```

```
## 290 0.6121673 1.00000000          lr
## 291 0.6098485 1.00000000          lr
## 292 0.6075472 1.00000000          lr
## 293 0.6052632 1.00000000          lr
## 294 0.6029963 1.00000000          lr
## 295 0.6007463 1.00000000          lr
## 296 0.5985130 1.00000000          lr
## 297 0.5962963 1.00000000          lr
## 298 0.5940959 1.00000000          lr
## 299 0.5919118 1.00000000          lr
## 300 0.5897436 1.00000000          lr
## 301 0.5875912 1.00000000          lr
## 302 0.5854545 1.00000000          lr
## 303 0.5833333 1.00000000          lr
## 304 0.5812274 1.00000000          lr
## 305 0.5791367 1.00000000          lr
## 306 0.5770609 1.00000000          lr
## 307 0.5750000 1.00000000          lr
## 308 0.5729537 1.00000000          lr
## 309 0.5709220 1.00000000          lr
## 310 0.5689046 1.00000000          lr
## 311 0.5669014 1.00000000          lr
## 312 0.5649123 1.00000000          lr
## 313 0.5629371 1.00000000          lr
## 314 0.5609756 1.00000000          lr
## 315 0.5590278 1.00000000          lr
## 316 0.5570934 1.00000000          lr
## 317 0.5551724 1.00000000          lr
## 318 0.5532646 1.00000000          lr
## 319 0.5513699 1.00000000          lr
## 320 0.5494881 1.00000000          lr
## 321 0.5476190 1.00000000          lr
## 322 0.5457627 1.00000000          lr
## 323 0.5439189 1.00000000          lr
## 324 0.5420875 1.00000000          lr
## 325 0.5402685 1.00000000          lr
## 326       NaN 0.00000000  tree.small
## 327 0.8928571 0.62111801  tree.small
## 328 0.8680556 0.77639752  tree.small
## 329 0.8305085 0.91304348  tree.small
## 330 0.7883598 0.92546584  tree.small
## 331 0.5698925 0.98757764  tree.small
## 332 0.5402685 1.00000000  tree.small
## 333       NaN 0.00000000 tree.medium
## 334 0.8928571 0.62111801 tree.medium
## 335 0.8680556 0.77639752 tree.medium
## 336 0.8421053 0.89440994 tree.medium
## 337 0.7883598 0.92546584 tree.medium
## 338 0.5698925 0.98757764 tree.medium
## 339 0.5402685 1.00000000 tree.medium
```

```
final.prec.recall.thresholds.df <- full_join(
    final.TPR.precision.50 %>%
        pivot_wider(names_from = 'metric', values_from = 'value_50') %>%
```

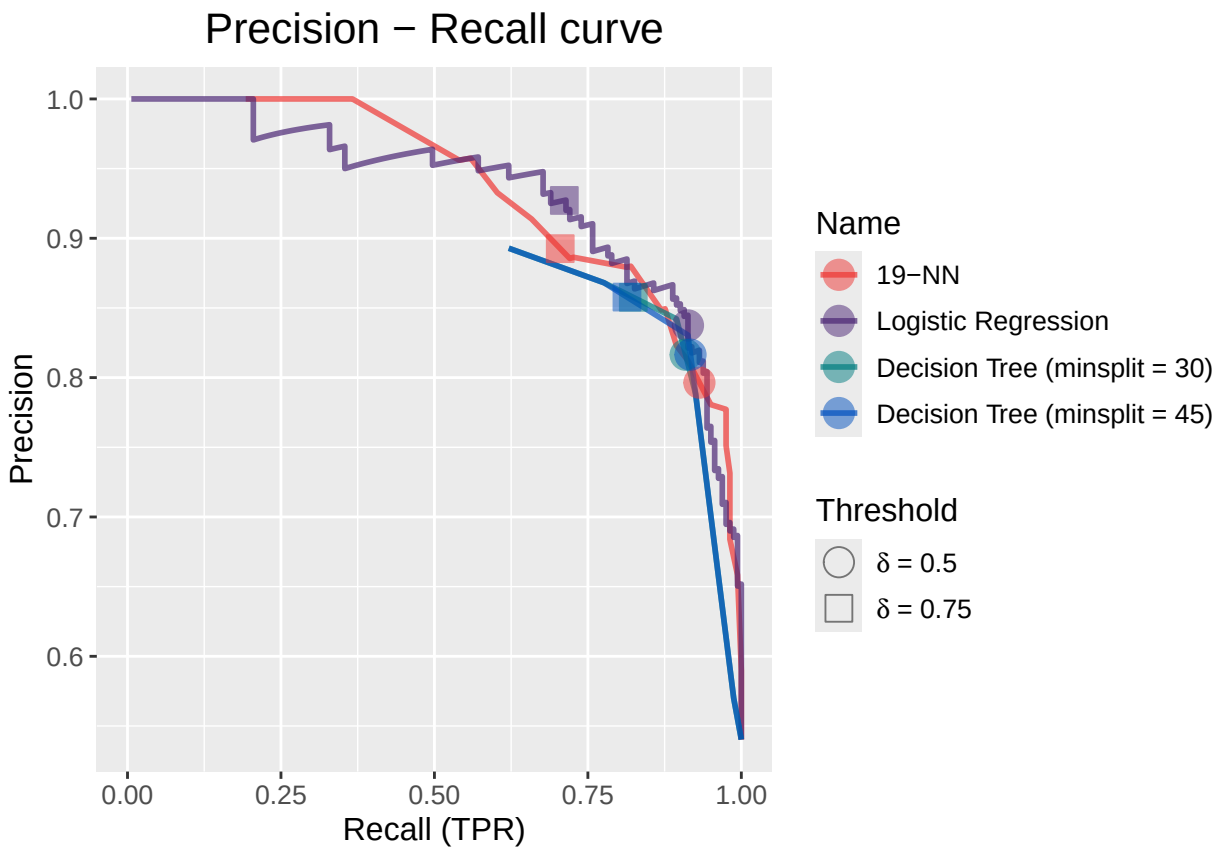
```
        mutate(threshold = 0.5),
    final.TPR.precision.75 %>%
        pivot_wider(names_from = 'metric', values_from = 'value_75') %>%
        mutate(threshold = 0.75),
)
```

```
## Joining with `by = join_by(Name, prec, tpr, threshold)`
```

```
(final.prec.recall.df %>% ggplot() +
    # facet_wrap(~ Name) +
    geom_line(
        mapping = aes(x = tpr, y = prec, col = Name),
        size = 1, alpha = .7
    ) +
    geom_point(
        data = final.prec.recall.thresholds.df,
        mapping = aes(
            x = tpr, y = prec, col = Name, fill = Name,
            pch = as.factor(threshold)
        ),
        size = 5,
        alpha = 0.5,
    ) +
        labs(
            x = 'Recall (TPR)',
            y = 'Precision',
            shape = 'Threshold'
        ) +
        scale_shape_manual(
            breaks = c('0.5', '0.75'),
            labels = c(
                TeX('\\delta = 0.5'),
                TeX('\\delta = 0.75')
            ),
            values = c(21, 22)
        ) +
        scale_color_manual(
            breaks = c(
                'knn', 'lr',
                'tree.medium',  'tree.small'
            ),
            labels = c(
                '19-NN', 'Logistic Regression',
                'Decision Tree (minsplit = 30)',  'Decision Tree (minsplit = 45)'
            ),
            values=c(
                '#ee3532', '#4a2574',
                '#007d80',  '#0050bf'
            )
        ) +
        scale_fill_manual(
            breaks = c(
                'knn', 'lr',
                'tree.medium',  'tree.small'
```

```
            ),
            labels = c(
                '19-NN', 'Logistic Regression',
                'Decision Tree (minsplit = 30)',  'Decision Tree (minsplit = 45)'
            ),
            values=c(
                '#ee3532', '#4a2574',
                '#007d80',  '#0050bf'
            )
        ) +
        theme(
            # legend.position = 'inside',
            # legend.direction = 'horizontal',
            # legend.box = 'horizontal',
            # legend.box.just = 'right',
            # legend.justification = c('right', 'bottom'),
            legend.background = element_blank(),
            strip.background = element_blank(),
            strip.text.x = element_blank()
        )
) %>%
        .save_and_display(
            'Precision - Recall curve',
            width = 960, height = 640,
            '../figures/40.Precision-Recall.pdf'
        )
```

```
## Warning: Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
## Removed 4 rows containing missing values or values outside the scale range
## (`geom_line()`).
```


Precision − Recall curve
Precision
1.0
0.9
0.8
0.7
0.6
0.00
0.25
0.50
0.75
1.00
Recall (TPR)
Name
19−NN
Logistic Regression
Decision Tree (minsplit = 30)
Decision Tree (minsplit = 45)
Threshold
δ = 0.5
δ = 0.75