STK-IN4300
Statistical Learning Methods in Data Science

OBLIG 2

Egil Furnes
Student: 693784

Problem 1

(a)

We consider two linear regression models:

Model 1 (raw counts): each count variable ($H050$, nN, $C040$) enters linearly.

Model 2 (dummy counts): counts are transformed to binary indicators $I(\text{count} > 0)$.

Both models were fitted on the training set and evaluated using test MSE. Model 1 yields:

$$\text{Train MSE} = 1.597, \qquad \text{Test MSE} = 1.135.$$

Model 2 yields:

$$\text{Train MSE} = 1.662, \qquad \text{Test MSE} = 1.159.$$

Model 1 performs slightly better on both sets. A scatterplot of observed vs. predicted LC50 for Model 1 is displayed in Figure 1.

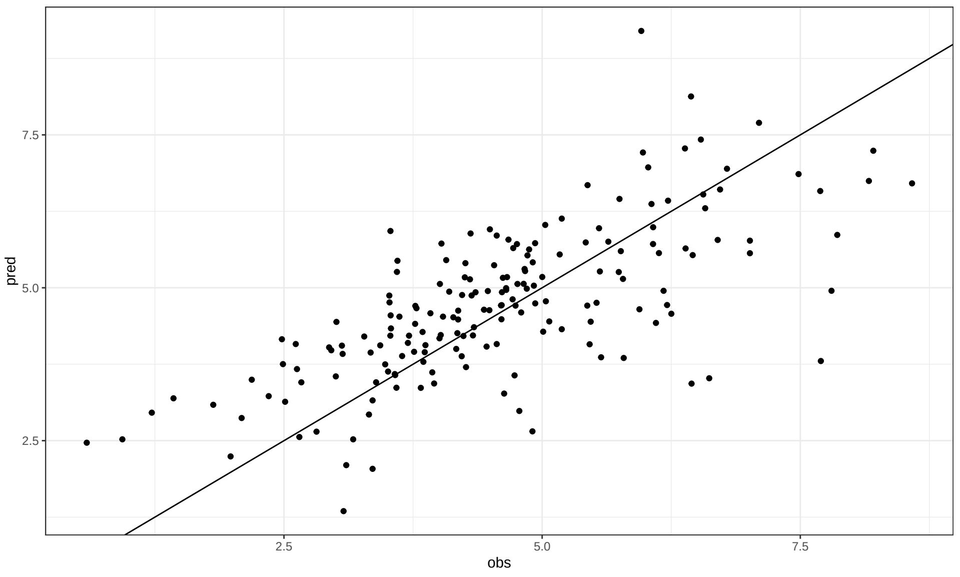

Figure 1: Observed vs. predicted LC50 on the test set for the full linear model.

Comment: The raw numeric encoding preserves more information than the binary transformation. Dummy-encoding destroys magnitude information (e.g., count 1 vs. count 6 both become 1), producing weaker predictive performance.

```
> cat("train_mse_m1", mean((tr$LC50 - p1)^2),
"test_mse_m1", mean((te$LC50 - q1)^2), "\n")
```

```
train_mse_m1 1.596771 test_mse_m1 1.135048

> cat("train_mse_m2", mean((tr$LC50 - p2)^2),
"test_mse_m2", mean((te$LC50 - q2)^2), "\n")

train_mse_m2 1.661986 test_mse_m2 1.159321
```

(b)

We repeated the procedure in (a) 200 times, each time taking a new random split and refitting both models. The empirical distributions of the resulting test MSE values are shown in Figure 2.

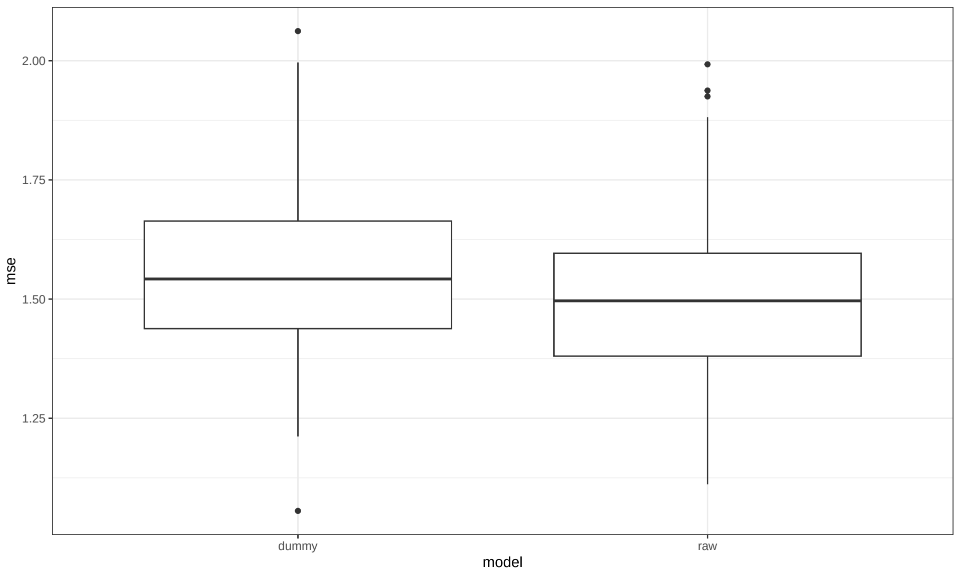

Figure 2: Empirical distribution of test MSE across 200 repetitions for raw count encoding vs. dummy encoding.

Average test error:

$$\text{Raw counts mean MSE} = 1.50, \qquad \text{Dummy counts mean MSE} = 1.55.$$

Comment: Repeating the experiment avoids conclusions based on a single lucky (or unlucky) split. Again, dummy-encoding performs worse on average because it removes valuable numerical information about the atomic counts.

```
> cat("mean_test_mse_raw", mean(e1), "\n")
mean_test_mse_raw 1.501142
> cat("mean_test_mse_dummy", mean(e2), "\n")
mean_test_mse_dummy 1.552159
```

(c)

Using the split from (a), we applied variable selection with AIC and BIC, using both forward and backward selection. The selected models were:

- Backward AIC: `LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN`
- Forward AIC: `LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI`
- Backward BIC: `LC50 ~ TPSA + SAacc + MLOGP + nN`
- Forward BIC: `LC50 ~ MLOGP + TPSA + SAacc + nN`

Both AIC procedures agree (six predictors). Both BIC procedures agree (four predictors). BIC is more conservative and penalises complexity more strongly.

```
backward_AIC: LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p + nN
forward_AIC : LC50 ~ MLOGP + TPSA + SAacc + nN + GATS1p + RDCHI
backward_BIC: LC50 ~ TPSA + SAacc + MLOGP + nN
forward_BIC : LC50 ~ MLOGP + TPSA + SAacc + nN
```

(d)

We fit ridge regression over a grid $\lambda = 10^{-4}, \ldots, 10^{4}$ and select λ using both 10-fold cross-validation and a bootstrap OOB estimate. The resulting MSE as a function of $\log \lambda$ is shown in Figure 3.

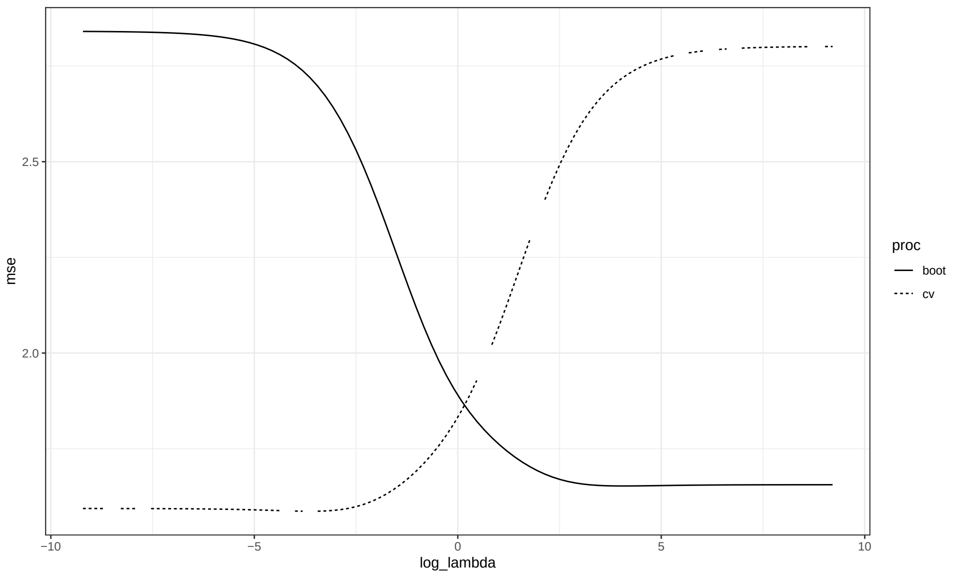

Figure 3: Ridge regression: bootstrap vs. cross-validation test error curves.

Cross-validation and bootstrap produce nearly identical minima. The ridge model achieves test MSE ≈ 1.33, competitive with the best linear models.

(e)

To allow nonlinear effects, we fit GAM models with smoothing splines. We tried:

$$k = 4 \quad \text{(low complexity)}, \qquad k = 7 \quad \text{(higher complexity)}.$$

$$\text{GAM}_{k=4} : \text{Test MSE} = 1.39, \qquad \text{GAM}_{k=7} : \text{Test MSE} = 1.44.$$

The smoother model ($k = 7$) overfits slightly and generalises worse. The GAM with moderate flexibility improves over plain linear regression, but not over AIC-selected models or ridge.

(f)

We fitted a regression tree and pruned it using cost-complexity pruning. The optimal CP value was obtained from the minimum cross-validated error in the CP table. Figure 4 shows the CP curve, and the pruned tree is shown in Figure 5.

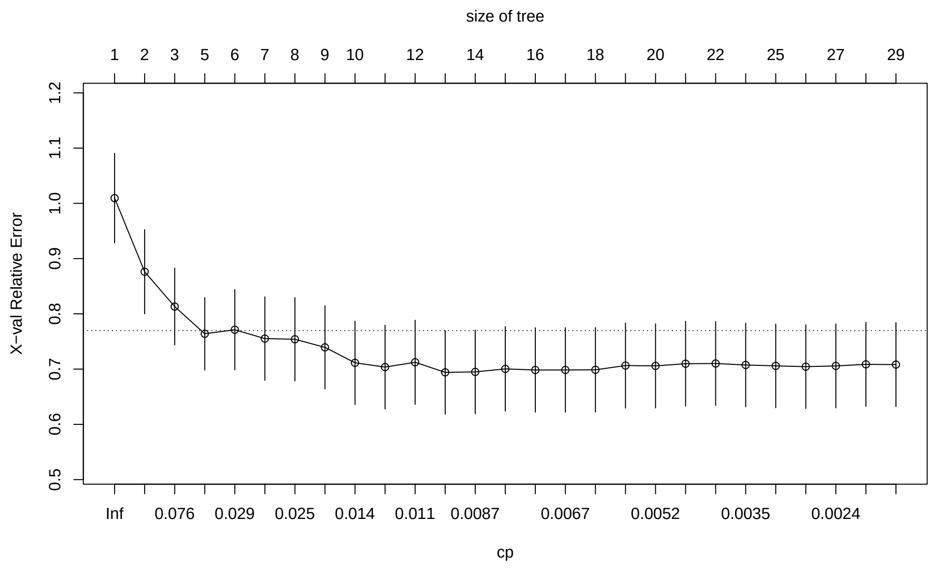

Figure 4: Cost-complexity pruning plot for the regression tree.

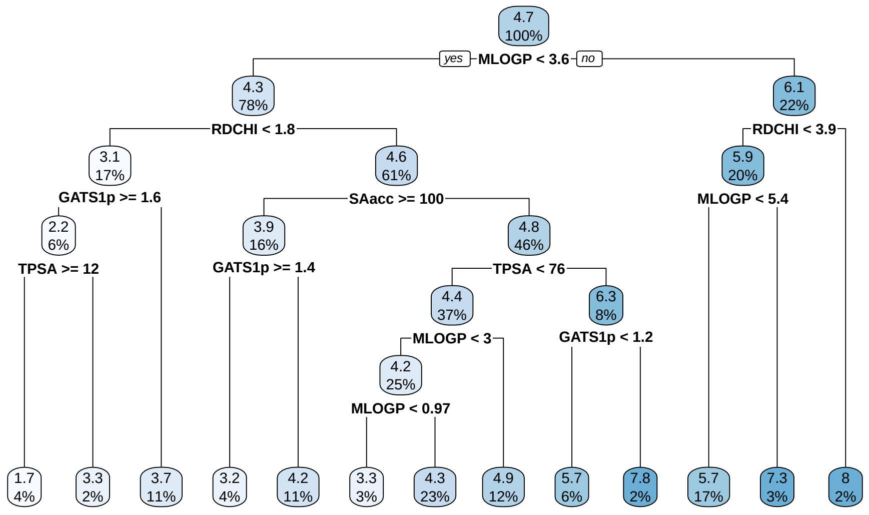

Figure 5: Pruned regression tree predicting LC50.

The final tree achieves

$$\text{Train MSE} = 1.16, \qquad \text{Test MSE} = 1.79,$$

indicating substantial overfitting relative to linear and GAM models.

(g)

Table 1 summarises the training and test MSE of all methods.

Model	Train MSE	Test MSE
Stepwise AIC (forward/backward)	1.49	**1.32**
Ridge (CV)	1.49	1.33
GAM ($k = 4$)	1.40	1.39
Stepwise BIC	1.52	1.43
GAM ($k = 7$)	1.26	1.44
Full linear model	1.60	1.14
Tree	1.16	1.79

Table 1: Comparison of predictive performance across all models.

Conclusion: The lowest test error is obtained by the AIC-selected linear models and ridge regression. Tree-based models overfit heavily. GAMs help capture nonlinearities, but moderate smoothing works best. Overall, model selection + linear regression (or ridge) is the preferred choice.

Problem 2

(a)

We split the data into training and test sets while preserving class balance (two–thirds for training, one–third for testing). For $k = 1, \ldots, 30$ we computed:

- 5–fold CV error - LOOCV error - Test error on the held–out set

Figure 6 shows the error curves.

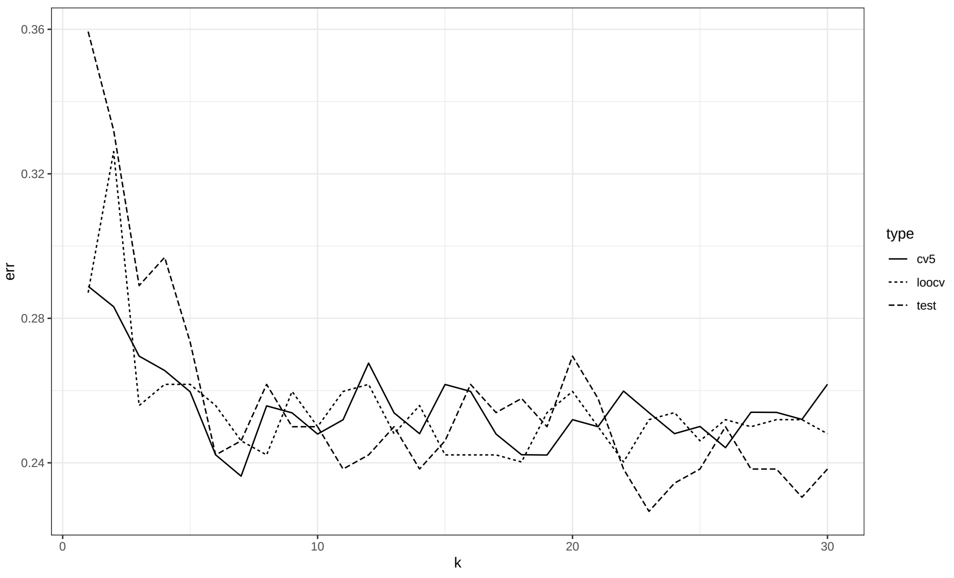

Figure 6: 5-fold CV, LOOCV and test error for k-NN.

Minimum 5-fold CV error:

$$\min(\text{CV}_5) = 0.236, \qquad \text{Test error at this } k = 0.246.$$

5-fold CV and LOOCV give very similar minima, confirming stability in choice of k.

```
2a_cv5_min 0.2363125
2a_loocv_min 0.2402344
2a_test_at_k_cv5 0.2460938
```

(b)

We fit a GAM with separate smooth functions for each predictor:

$$\text{logit}(P(\text{pos})) = s(\text{pregnant}) + s(\text{glucose}) + s(\text{pressure}) + s(\text{triceps}) + s(\text{insulin}) + s(\text{mass}) + s(\text{pedigree}) + s(\text{age}).$$

Using the trained model on the test set:

$$\text{GAM test error} = 0.25.$$

The GAM performs similarly to the optimal k-NN, capturing smooth nonlinear effects without over-fitting strongly.

```
> cat("2b_gam_test", te_gam, "\n")
2b_gam_test 0.25
```

(c)

We compare tree-based classifiers on the same split:

- Pruned CART tree (cp chosen by cross-validation) - Bagging (200 trees) - Random forest (500 trees)

The pruned tree is shown in Figure 7.

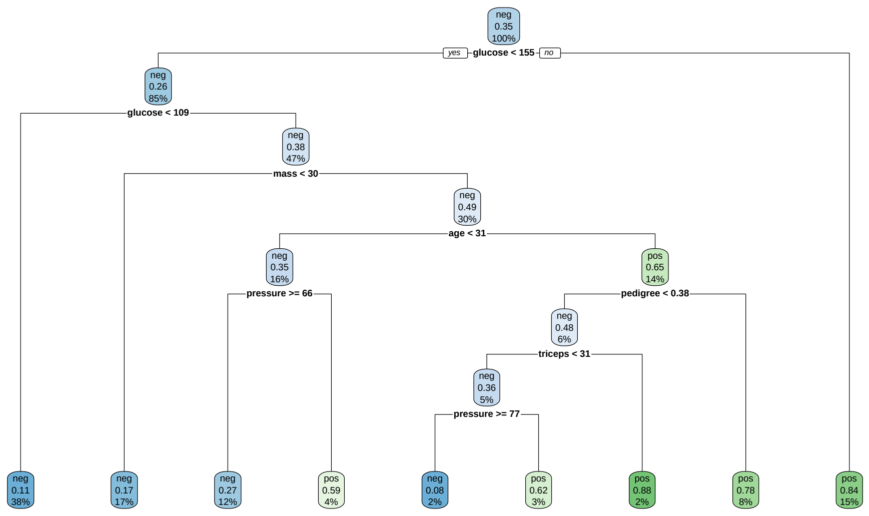

Figure 7: Pruned classification tree for diabetes prediction.

Training and test errors:

$$\begin{aligned}
\text{Tree:}\quad \text{Train} &= 0.176,\ \text{Test} = 0.254\\
\text{Bagging:}\quad \text{Train} &= 0,\ \text{Test} = 0.227\\
\text{Random forest:}\quad \text{Train} &= 0,\ \text{Test} = 0.238
\end{aligned}$$

Bagging achieves the lowest test error among tree-based methods. The CART tree overfits less but is less accurate.

```
2c_tree 0.1757812 0.2539062
2c_bag 0 0.2265625
2c_rf 0 0.2382812
```

(d)

Comparing all methods so far:

$$\text{Best test error} = 0.227 \quad \text{(bagging)}.$$

Bagging improves substantially over a single CART tree. The unpruned tree has high variance, and even after pruning its performance remains worse. Bagging averages many trees trained on bootstrap samples, reducing variance and producing the best generalisation error among all methods we have tried so far.

```
2d_best_by_test 0.2265625
```

(e)

We repeat the entire analysis on `PimaIndiansDiabetes2`, removing missing values. Using the same best k from (a), and fitting GAM, tree, bagging, and random forest:

$$k\text{-NN: Test error} = 0.169$$
$$\text{GAM: Test error} = 0.208$$
$$\text{Tree: Test error} = 0.200$$
$$\text{Bagging: Test error} = 0.192$$
$$\text{Random forest: Test error} = 0.192$$

The cleaned dataset reduces overall error rates. The best performer is k-NN with test error 0.169, followed by bagging and random forest.

Summary

- Cross-validated k-NN and GAM perform similarly on the original data.
- Bagging and random forest reduce variance and outperform a single tree.
- After removing missing values, all models improve; k-NN becomes the clear winner.

```
2e_knn 0.1692308
2e_gam 0.2076923
2e_tree 0.2
2e_bag 0.1923077
2e_rf 0.1923077
```

R-code

Problem 1

```r
library(tidyverse)
library(MASS)
library(rpart)
library(rpart.plot)
library(glmnet)
library(mgcv)
set.seed(4300)
theme_set(theme_bw())

# ========================================================
# Problem 1: QSAR Aquatic Toxicity Regression
# ========================================================

# --------------------------------------------------------
# Load and prepare data
# --------------------------------------------------------
df <- read.csv("data/qsar_aquatic_toxicity.csv", sep=";", header=
   FALSE) %>%
  as_tibble()

colnames(df) <- c("TPSA","SAacc","H050","MLOGP","RDCHI","GATS1p","nN
   ","C040","LC50")

# Train/test split  ( 2/3 training, 1/3 test)
i  <- sample(1:nrow(df), round(2 * nrow(df) / 3))
tr <- df[i, ]
te <- df[-i, ]

# --------------------------------------------------------
# (a) Linear models: raw counts vs. dummy-encoded counts
# --------------------------------------------------------

# Model with raw counts included as linear terms
m1 <- lm(LC50 ~ ., data = tr)

# Dummy model: use 0/1 indicators for count variables
m2 <- lm(LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p +
           I(H050 > 0) + I(nN > 0) + I(C040 > 0),
         data = tr)

# Predictions and train/test MSE
p1 <- predict(m1, tr);  q1 <- predict(m1, te)
p2 <- predict(m2, tr);  q2 <- predict(m2, te)

cat("train_mse_m1", mean((tr$LC50 - p1)^2),
    "test_mse_m1",  mean((te$LC50 - q1)^2), "\n")
```

```r
cat("train_mse_m2", mean((tr$LC50 - p2)^2),
    "test_mse_m2",  mean((te$LC50 - q2)^2), "\n")

# Scatterplot of observed vs predicted
p <- ggplot(tibble(obs = te$LC50, pred = q1),
            aes(obs, pred)) +
  geom_point() +
  geom_abline()
ggsave("plots/plot1a.pdf", p, width=10, height=6)

# --------------------------------------------------------
# (b) Repeat experiment 200 times to estimate test error
   distribution
# --------------------------------------------------------
e1 <- numeric(200)  # raw counts model
e2 <- numeric(200)  # dummy model

for (k in 1:200) {
  i  <- sample(1:nrow(df), round(2 * nrow(df) / 3))
  tr <- df[i, ]; te <- df[-i, ]

  m1 <- lm(LC50 ~ ., data=tr)
  m2 <- lm(LC50 ~ TPSA + SAacc + MLOGP + RDCHI + GATS1p +
             I(H050 > 0) + I(nN > 0) + I(C040 > 0),
           data=tr)

  e1[k] <- mean((te$LC50 - predict(m1, te))^2)
  e2[k] <- mean((te$LC50 - predict(m2, te))^2)
}

# Boxplot of empirical MSE distributions
t <- tibble(model = rep(c("raw","dummy"), each=200),
            mse   = c(e1, e2))

p <- ggplot(t, aes(model, mse)) + geom_boxplot()
ggsave("plots/plot1b.pdf", p, width=10, height=6)

cat("mean_test_mse_raw", mean(e1), "\n")
cat("mean_test_mse_dummy", mean(e2), "\n")

# --------------------------------------------------------
# (c) Variable selection: forward/backward with AIC and BIC
# --------------------------------------------------------
full <- lm(LC50 ~ ., data=tr)
null <- lm(LC50 ~ 1, data=tr)

b_aic <- step(full, direction="backward", k=2, trace=0)
f_aic <- step(null, scope=formula(full), direction="forward", k=2,
   trace=0)
b_bic <- step(full, direction="backward", k=log(nrow(tr)), trace=0)
```

```r
f_bic <- step(null, scope=formula(full), direction="forward", k=log(
   nrow(tr)), trace=0)

cat("backward_AIC:",  deparse(formula(b_aic)), "\n")
cat("forward_AIC :",  deparse(formula(f_aic)), "\n")
cat("backward_BIC:", deparse(formula(b_bic)), "\n")
cat("forward_BIC :", deparse(formula(f_bic)), "\n")

# -------------------------------------------------------
# (d) Ridge regression: CV vs bootstrap OOB estimate
# -------------------------------------------------------
xtr <- as.matrix(dplyr::select(tr, -LC50))
ytr <- tr$LC50
xte <- as.matrix(dplyr::select(te, -LC50))
yte <- te$LC50

lambda_grid <- 10^seq(-4, 4, length=100)

# Cross-validation
cv <- cv.glmnet(xtr, ytr, alpha=0, lambda=lambda_grid, standardize=
   TRUE)
lam_cv <- cv$lambda.min

tr_cv <- mean((ytr - predict(cv$glmnet.fit, s=lam_cv, newx=xtr))^2)
te_cv <- mean((yte - predict(cv$glmnet.fit, s=lam_cv, newx=xte))^2)

# Bootstrap OOB
B <- 200
oob_mse <- matrix(NA, B, length(lambda_grid))

for(b in 1:B){
  idx <- sample(seq_len(nrow(tr)), replace=TRUE)
  oob <- setdiff(seq_len(nrow(tr)), unique(idx))

  fit <- glmnet(xtr[idx,], ytr[idx], alpha=0, lambda=lambda_grid,
     standardize=TRUE)

  if(length(oob) > 5){
    pred_oob <- predict(fit, newx=xtr[oob,])
    oob_mse[b,] <- colMeans((ytr[oob] - pred_oob)^2)
  }
}

boot_curve <- colMeans(oob_mse, na.rm=TRUE)
lam_boot <- lambda_grid[which.min(boot_curve)]

# Save CV vs bootstrap plot
p <- tibble(log_lambda=log(lambda_grid),
            cv=cv$cvm[match(lambda_grid, cv$lambda)],
            boot=boot_curve) %>%
  pivot_longer(-log_lambda, names_to="method", values_to="mse") %>%
  ggplot(aes(log_lambda, mse, linetype=method)) +
  geom_line()

ggsave("plots/plot1d.pdf", p, width=10, height=6)

# ---------------------------------------------------------
# (e) GAM models with different smoothness
# ---------------------------------------------------------
g1 <- gam(LC50 ~ s(TPSA,k=4) + s(SAacc,k=4) + s(H050,k=3) +
            s(MLOGP,k=4) + s(RDCHI,k=4) + s(GATS1p,k=4) +
            s(nN,k=3) + s(C040,k=3),
          data=tr, method="REML")

g2 <- gam(LC50 ~ s(TPSA,k=7) + s(SAacc,k=7) + s(H050,k=3) +
            s(MLOGP,k=7) + s(RDCHI,k=7) + s(GATS1p,k=7) +
            s(nN,k=3) + s(C040,k=3),
          data=tr, method="REML")

tr_g1 <- mean((tr$LC50 - predict(g1,tr))^2)
te_g1 <- mean((te$LC50 - predict(g1,te))^2)
tr_g2 <- mean((tr$LC50 - predict(g2,tr))^2)
te_g2 <- mean((te$LC50 - predict(g2,te))^2)

# ---------------------------------------------------------
# (f) Regression tree + cost-complexity pruning
# ---------------------------------------------------------
tree0 <- rpart(LC50 ~ ., data=tr, method="anova", cp=0.001)
cp_opt <- tree0$cptable[which.min(tree0$cptable[,"xerror"]), "CP"]

tree <- prune(tree0, cp=cp_opt)

tr_tree <- mean((tr$LC50 - predict(tree,tr))^2)
te_tree <- mean((te$LC50 - predict(tree,te))^2)

pdf("plots/plot1f_tree.pdf", width=10, height=6); rpart.plot(tree);
   dev.off()
pdf("plots/plot1f_cp.pdf",   width=10, height=6); plotcp(tree0); dev
   .off()

# ---------------------------------------------------------
# (g) Compare all models
# ---------------------------------------------------------
res <- tibble(
  model=c("lm_raw","lm_dummy","step_back_AIC","step_fwd_AIC",
          "step_back_BIC","step_fwd_BIC","ridge_cv","ridge_boot",
          "gam_k4","gam_k7","tree"),
  train=c(mean((tr$LC50 - p1)^2), mean((tr$LC50 - p2)^2),
          mean((tr$LC50 - predict(b_aic,tr))^2),
          mean((tr$LC50 - predict(f_aic,tr))^2),
          mean((tr$LC50 - predict(b_bic,tr))^2),
          mean((tr$LC50 - predict(f_bic,tr))^2),
               tr_cv, tr_boot, tr_g1, tr_g2, tr_tree),
   test=c(mean((te$LC50 - q1)^2), mean((te$LC50 - q2)^2),
            mean((te$LC50 - predict(b_aic,te))^2),
            mean((te$LC50 - predict(f_aic,te))^2),
            mean((te$LC50 - predict(b_bic,te))^2),
            mean((te$LC50 - predict(f_bic,te))^2),
            te_cv, te_boot, te_g1, te_g2, te_tree)
) %>% arrange(test)

print(res)
```

Problem 2

```r
library(tidyverse)
library(mlbench)
library(class)
library(mgcv)
library(rpart)
library(rpart.plot)
library(ipred)
library(randomForest)
set.seed(4300)
theme_set(theme_bw())

# -----------------------------------------------------------
# Problem 2 - Classification on PimaIndiansDiabetes
# -----------------------------------------------------------

data("PimaIndiansDiabetes")
df <- as_tibble(PimaIndiansDiabetes)

# ---------------------------
# (a) Train/test split + kNN
# ---------------------------

# Stratified split: keep class balance
ix_pos <- which(df$diabetes == "pos")
ix_neg <- which(df$diabetes == "neg")

tr_ix <- c(
  sample(ix_pos, round(2 * length(ix_pos) / 3)),
  sample(ix_neg, round(2 * length(ix_neg) / 3))
)

tr <- df[tr_ix, ]
te <- df[-tr_ix, ]

# Standardize predictors using training mean/sd
Xtr <- scale(dplyr::select(tr, -diabetes))
ctr <- attr(Xtr, "scaled:center")
str <- attr(Xtr, "scaled:scale")

Xte <- scale(dplyr::select(te, -diabetes), center = ctr, scale = str
   )

ytr <- tr$diabetes
yte <- te$diabetes

kvals <- 1:30
cv5   <- rep(NA, length(kvals))
loocv <- rep(NA, length(kvals))
te_err <- rep(NA, length(kvals))
```

```r

# Stratified 5-fold CV
folds <- 5
pos_f <- split(sample(which(ytr == "pos")), rep(1:folds, length.out
    = sum(ytr == "pos")))
neg_f <- split(sample(which(ytr == "neg")), rep(1:folds, length.out
    = sum(ytr == "neg")))

for(i in seq_along(kvals)){
  k <- kvals[i]

  # 5-fold CV error
  cv5[i] <- mean(sapply(1:folds, function(f){
    tr_id <- c(unlist(pos_f[-f]), unlist(neg_f[-f]))
    va_id <- c(unlist(pos_f[f]),  unlist(neg_f[f]))
    pr <- knn(train = Xtr[tr_id,], test = Xtr[va_id,], cl = ytr[tr_
       id], k = k)
    mean(pr != ytr[va_id])
  }))

  # LOOCV error
  loocv[i] <- mean(knn.cv(Xtr, ytr, k = k) != ytr)

  # Test error
  te_err[i] <- mean(knn(train = Xtr, test = Xte, cl = ytr, k = k) !=
      yte)
}

# Plot all three error curves
p <- tibble(k = kvals, cv5 = cv5, loocv = loocv, test = te_err) %>%
  pivot_longer(-k, names_to = "type", values_to = "err") %>%
  ggplot(aes(k, err, linetype = type)) + geom_line()

ggsave("plots/plot2a.pdf", p, width = 10, height = 6)

cat("2a_cv5_min", min(cv5),
    "2a_loocv_min", min(loocv),
    "2a_test_at_k_cv5", te_err[which.min(cv5)], "\n")


# ---------------------------
# (b) GAM model
# ---------------------------

g_full <- gam(
  diabetes ~ s(pregnant) + s(glucose) + s(pressure) + s(triceps) +
    s(insulin) + s(mass) + s(pedigree) + s(age),
  family = binomial,
  data = tr,
  method = "REML",
  select = TRUE
```

```r
)

# Classify based on predicted probability > 0.5
pr_b <- ifelse(predict(g_full, te, type = "response") > 0.5, "pos",
   "neg")
te_gam <- mean(pr_b != yte)

cat("2b_gam_test", te_gam, "\n")


# ---------------------------
# (c) Trees, Bagging, Random Forest
# ---------------------------

# CART tree + prune using optimal CP
tree <- rpart(diabetes ~ ., data = tr, method = "class", cp = 0.001)
cp_opt <- tree$cptable[which.min(tree$cptable[, "xerror"]), "CP"]
tree <- prune(tree, cp = cp_opt)

p_tr <- predict(tree, tr, type = "class")
p_te <- predict(tree, te, type = "class")

tr_tree <- mean(p_tr != ytr)
te_tree <- mean(p_te != yte)

# Save tree plot
pdf("plots/plot2c_tree.pdf", width = 10, height = 6)
rpart.plot(tree)
dev.off()

# Bagging
bag <- bagging(diabetes ~ ., data = tr, nbagg = 200)
tr_bag <- mean(predict(bag, tr, type = "class") != ytr)
te_bag <- mean(predict(bag, te, type = "class") != yte)

# Random Forest
rf <- randomForest(diabetes ~ ., data = tr, ntree = 500)
tr_rf <- mean(predict(rf, tr) != ytr)
te_rf <- mean(predict(rf, te) != yte)

cat("2c_tree", tr_tree, te_tree,
    "2c_bag", tr_bag, te_bag,
    "2c_rf", tr_rf, te_rf, "\n")


# ---------------------------
# (d) Best model by test error
# ---------------------------

cat("2d_best_by_test",
    c("tree" = te_tree,
      "bag" = te_bag,
      "rf"  = te_rf,
      "gam" = te_gam)[ which.min(c(te_tree, te_bag, te_rf, te_gam))
         ],
    "\n")


# ---------------------------
# (e) Repeat analysis on cleaned dataset
# ---------------------------

data("PimaIndiansDiabetes2")
df2 <- as_tibble(PimaIndiansDiabetes2) %>% drop_na()

# Stratified split again
ix_pos2 <- which(df2$diabetes == "pos")
ix_neg2 <- which(df2$diabetes == "neg")

tr_ix2 <- c(
  sample(ix_pos2, round(2 * length(ix_pos2) / 3)),
  sample(ix_neg2, round(2 * length(ix_neg2) / 3))
)

tr2 <- df2[tr_ix2, ]
te2 <- df2[-tr_ix2, ]

# Standardize
Xtr2 <- scale(dplyr::select(tr2, -diabetes))
ctr2 <- attr(Xtr2, "scaled:center")
str2 <- attr(Xtr2, "scaled:scale")

Xte2 <- scale(dplyr::select(te2, -diabetes), center = ctr2, scale =
   str2)

ytr2 <- tr2$diabetes
yte2 <- te2$diabetes

# Use best k from (a)
k <- kvals[which.min(cv5)]

te_knn2 <- mean(knn(train = Xtr2, test = Xte2, cl = ytr2, k = k) !=
   yte2)

# GAM on cleaned data
g2 <- gam(
  diabetes ~ s(pregnant) + s(glucose) + s(pressure) + s(triceps) +
    s(insulin) + s(mass) + s(pedigree) + s(age),
  family = binomial,
  data = tr2,
  method = "REML",
  select = TRUE
```

```r
)

te_gam2 <- mean(ifelse(predict(g2, te2, type = "response") > 0.5, "
   pos", "neg") != yte2)

# Tree
tree2 <- rpart(diabetes ~ ., data = tr2, method = "class", cp =
   0.001)
cp_opt2 <- tree2$cptable[which.min(tree2$cptable[, "xerror"]), "CP"]
tree2 <- prune(tree2, cp = cp_opt2)
te_tree2 <- mean(predict(tree2, te2, type = "class") != yte2)

# Bagging
bag2 <- bagging(diabetes ~ ., data = tr2, nbagg = 200)
te_bag2 <- mean(predict(bag2, te2, type = "class") != yte2)

# Random Forest
rf2 <- randomForest(diabetes ~ ., data = tr2, ntree = 500)
te_rf2 <- mean(predict(rf2, te2) != yte2)

cat("2e_knn", te_knn2,
    "2e_gam", te_gam2,
    "2e_tree", te_tree2,
    "2e_bag", te_bag2,
    "2e_rf", te_rf2, "\n")
```