

C. code

```
library(readxl)
library(arsenal)
library(glmnet)
library(scales)
library(xtable)
library(kableExtra)
library(parallel)
library(foreach)
library(doParallel)

svm.cv.mine <- function(q3.train, form, k, response_variable) {
  k = k
  folds = caret::createFolds(1:nrow(q3.train), k = k)

  gamma.seq = seq(from = 1e-04, to = 0.01, length.out = 50)
  cost = seq(from = 0.1, to = 5, length.out = 50)
  numeric.paras = expand.grid(1:50, 1:50)
  params.list = c(list(list(kernel = "linear")), lapply(1:50^2,
    function(i) {
      list(kernel = "polynomial", gamma = gamma.seq[numeric.paras[i,
        1]], degree = 2, cost = cost[numeric.paras[i,
        2]])
    }), lapply(1:50^2, function(i) {
      list(kernel = "polynomial", gamma = gamma.seq[numeric.paras[i,
        1]], cost = cost[numeric.paras[i, 2]], degree = 3)
    }), lapply(1:50^2, function(i) {
      list(kernel = "polynomial", gamma = gamma.seq[numeric.paras[i,
        1]], cost = cost[numeric.paras[i, 2]], degree = 4)
    }), lapply(1:50^2, function(i) {
      list(kernel = "radial", gamma = gamma.seq[numeric.paras[i,
        1]], cost = cost[numeric.paras[i, 2]])
    }), lapply(1:50^2, function(i) {
      list(kernel = "sigmoid", gamma = gamma.seq[numeric.paras[i,
        1]], cost = cost[numeric.paras[i, 2]])
    })))

  numCores <- as.integer(future::availableCores()) - 1
  registerDoParallel(numCores)

  q3e.val.error = foreach(x = params.list) %dopar% {
    library(e1071)
    validation_error = numeric(k)
    for (i in 1:k) {
      val.train = q3.train[-folds[[i]], ]
      val.test = q3.train[folds[[i]], ]
    }
  }
```

```

        fit = do.call(svm, c(list(formula = form, data = val.train),
                               x))
        predict.test = predict(fit, val.test, decision.values = F)

        validation_error[i] = mean(predict.test != val.test[[response_variable]])
      }
      return(mean(validation_error))
    }
  }

svm.cv.cost <- function(q3.train, form, k, response_variable,
                        cost.list) {

  n = length(cost.list)
  folds = caret::createFolds(1:nrow(q3.train), k = k)

  val.err.cost = numeric(n)
  for (j in 1:n) {
    validation_error = numeric(k)
    for (i in 1:k) {
      val.train = q3.train[-folds[[i]], ]
      val.test = q3.train[folds[[i]], ]

      fit = svm(form, data = val.train, cost = cost.list[j],
                 kernel = "linear")
      predict.test = predict(fit, val.test, decision.values = F)

      validation_error[i] = mean(predict.test != val.test[[response_variable]])
    }
    val.err.cost[j] = mean(validation_error)
  }
  return(val.err.cost)
}

svm.cv.cost.reg <- function(q3.train, form, k, response_variable,
                            cost.list) {

  n = length(cost.list)
  folds = caret::createFolds(1:nrow(q3.train), k = k)

  val.err.cost = numeric(n)
  for (j in 1:n) {
    validation_error = numeric(k)
    for (i in 1:k) {
      val.train = q3.train[-folds[[i]], ]
      val.test = q3.train[folds[[i]], ]

      fit = svm(form, data = val.train, cost = cost.list[j],
                 kernel = "linear")
      predict.test = predict(fit, val.test, decision.values = F)

      validation_error[i] = mean((predict.test - val.test[[response_variable]])^2)
    }
  }
}

```

```

    val.err.cost[j] = mean(validation_error)
  }
  return(val.err.cost)
}

correlation_test <- function(x1, x2, type = "chisq") {
  type1 = ifelse(length(unique(x1)) == 2, "binary", "continuous")
  type2 = ifelse(length(unique(x2)) == 2, "binary", "continuous")

  if (type1 == "binary" & type2 == "binary") {
    if (type == "chisq") {
      return(chisq.test(x1, x2, simulate.p.value = T)$p.value)
    }
    if (type == "fisher") {
      return(fisher.test(x1, x2)$p.value)
    }
  }
  if (type1 == "continuous" & type2 == "continuous") {
    return(summary(lm(x1 ~ x2))$coefficients[2, 4])
  }
  if (type1 == "categorical" & type2 == "continuous") {
    x3 = x1
    x1 = x2
    x2 = x3
  }
  x2 = as.factor(x2)
  x2 = c(0, 1)[as.numeric(x2)]
  return(t.test(x1 * x2, x1 * (1 - x2))$p.value)
}

library(ggplot2)
library(fastDummies)
library(tidyverse)
library(caret)
library(MASS)

par(mar = c(6, 6, 4, 4))

# read data -----
math = read.csv("student-mat.csv", sep = ";", stringsAsFactors = T)
str(math)
por = read.csv("student-por.csv", sep = ";", stringsAsFactors = T)
str(por)

# set dummies
math = fastDummies::dummy_columns(math, remove_first_dummy = T,
  remove_selected_columns = T)
por = fastDummies::dummy_columns(por, remove_first_dummy = T,
  remove_selected_columns = T)

predictors = setdiff(colnames(math), c("G1", "G2", "G3"))

```

```

math$grade.cat = cut(math$G1, breaks = c(-1, 9, 11, 13, 15, 20),
  labels = c("F", "D", "C", "B", "A"))
por$grade.cat = cut(por$G1, breaks = c(-1, 9, 11, 13, 15, 20),
  labels = c("F", "D", "C", "B", "A"))
por$grade.con = por$G1
math$grade.con = math$G1

# split data -----
set.seed(114514)
train_idx.math = sample(1:nrow(math), floor(nrow(math)) * 0.8,
  replace = F)
train_idx.por = sample(1:nrow(por), floor(nrow(por)) * 0.8, replace = F)

math.train = math[train_idx.math, ]
math.test = math[-train_idx.math, ]

por.train = por[train_idx.por, ]
por.test = por[-train_idx.por, ]

preproc.param <- math.train %>%
  preProcess(method = c("center", "scale"))
math.train <- preproc.param %>%
  predict(math.train)
math.test <- preproc.param %>%
  predict(math.test)
preproc.param <- por.train %>%
  preProcess(method = c("center", "scale"))
por.train <- preproc.param %>%
  predict(por.train)
por.test <- preproc.param %>%
  predict(por.test)

# EDA -----
response_variable = ""
# descriptive(sheet_name = '', data = math.train,
# descriptive_variables = colnames(math.train))

math.grade.train = math.train[, c("G1", "G2", "G3")]
por.grade.train = por.train[, c("G1", "G2", "G3")]

library(PerformanceAnalytics)

chart.Correlation(math.grade.train, histogram = TRUE, method = "pearson")
chart.Correlation(por.grade.train, histogram = TRUE, method = "pearson")

## correlation matrix -----
p = length(predictors)
math.cor.matrix = matrix(nrow = length(predictors), ncol = length(predictors))
por.cor.matrix = matrix(nrow = p, ncol = p)
for (i in 1:p) {
  for (j in 1:p) {
    math.cor.matrix[i, j] = correlation_test(math.train[[predictors[i]]],
      math.train[[predictors[j]]])
  }
}

```

```

      por.cor.matrix[i, j] = correlation_test(por.train[[predictors[i]]],
      por.train[[predictors[j]]])
    }
  }

library(reshape2)
library(ggplot2)

longData <- melt(math.cor.matrix)
longData <- melt(por.cor.matrix)

ggplot(longData, aes(x = Var2, y = Var1)) + geom_raster(aes(fill = value)) +
  scale_fill_gradient(low = "grey90", high = "red") + labs(x = "predictors",
  y = "predictors", title = "Matrix") + theme_bw() + theme(axis.text.x = element_text(size = 20,
  angle = 0, vjust = 0.3), axis.text.y = element_text(size = 20),
  plot.title = element_text(size = 20))

temp = longData$value
longData$value[which(temp < 0.05)] = 1
longData$value[which(temp >= 0.05)] = 0

ggplot(longData, aes(x = Var2, y = Var1)) + geom_raster(aes(fill = value)) +
  scale_fill_gradient(low = "grey90", high = "red") + labs(x = "predictors",
  y = "predictors", title = "Hypothesis test for independence between predictors") +
  theme_bw() + theme(axis.text.x = element_text(size = 20,
  angle = 0, vjust = 0.3), axis.text.y = element_text(size = 20),
  plot.title = element_text(size = 20), axis.title.x = element_text(size = 20),
  axis.title.y = element_text(size = 20), legend.position = "none")

# Classification ----- svm
# -----
library(e1071)
classification.formula = as.formula(paste("grade.cat~", paste(predictors,
collapse = "+"), sep = ""))
regression.formula = as.formula(paste("grade.con~", paste(predictors,
collapse = "+"), sep = ""))

math.svm.fit = svm(classification.formula, data = math.train)
math.svm.predict = predict(math.svm.fit, math.test)

### svm cv -----
set.seed(114514)
cost.list = exp(seq(from = -8, to = 0, length.out = 100))
svm.cv.val.error = svm.cv.cost(q3.train = math.train, form = classification.formula,
k = 10, response_variable = "grade.cat", cost.list = cost.list)
ggplot() + geom_line(aes(x = log(cost.list), y = svm.cv.val.error),
linewidth = 1.5) + theme(axis.text.x = element_text(size = 20,

```

```

    angle = 0, vjust = 0.3), axis.text.y = element_text(size = 20),
    plot.title = element_text(size = 20), axis.title.x = element_text(size = 20),
    axis.title.y = element_text(size = 20), legend.position = "none")

math.svm.fit = svm(classification.formula, data = math.train,
    cost = cost.list[which.min(svm.cv.val.error)], kernel = "linear")
math.svm.error.train = mean(predict(math.svm.fit) != math.train$grade.cat)
math.svm.error.test = mean(predict(math.svm.fit, math.test) !=
    math.test$grade.cat)

set.seed(114514)
cost.list = exp(seq(from = -8, to = 0, length.out = 100))
por.svm.cv.val.error = svm.cv.cost(q3.train = por.train, form = classification.formula,
    k = 10, response.variable = "grade.cat", cost.list = cost.list)
ggplot() + geom_line(aes(x = log(cost.list), y = por.svm.cv.val.error),
    linewidth = 1.5) + theme(axis.text.x = element_text(size = 20,
    angle = 0, vjust = 0.3), axis.text.y = element_text(size = 20),
    plot.title = element_text(size = 20), axis.title.x = element_text(size = 20),
    axis.title.y = element_text(size = 20), legend.position = "none")

por.svm.fit = svm(classification.formula, data = por.train, cost = cost.list[which.min(por.svm.cv.val.e
    kernel = "linear")
por.svm.error.train = mean(predict(por.svm.fit) != por.train$grade.cat)
por.svm.error.test = mean(predict(por.svm.fit, por.test) != por.test$grade.cat)

## logistic regression----- logistic regression
## -----
library(nnet)

math.logistic.fit = multinom(formula = classification.formula,
    data = math.train, Hess = T)
predict(math.logistic.fit)
summary(math.logistic.fit)
math.logistic.error.train = mean(math.train$grade.cat != predict(math.logistic.fit))
math.logistic.error.test = mean(math.test$grade.cat != predict(math.logistic.fit,
    math.test))

por.logistic.fit = multinom(formula = classification.formula,
    data = por.train, Hess = T)
predict(por.logistic.fit)
summary(por.logistic.fit)
por.logistic.error.train = mean(por.train$grade.cat != predict(por.logistic.fit))
por.logistic.error.test = mean(por.test$grade.cat != predict(por.logistic.fit,
    por.test))

## logistic regression lasso-----

### l-r LASSO math-----

```

```

set.seed(114514)
math.logistic.lasso.cv = cv.glmnet(x = as.matrix(math.train[,
  predictors]), y = math.train$grade.cat, family = "multinomial",
  alpha = 1, lambda = exp(seq(from = -5, to = 5, length.out = 100)),
  standardize = TRUE)
plot(math.logistic.lasso.cv, cex.lab = 2, cex.axis = 2)
math.logistic.lasso.best = glmnet(x = as.matrix(math.train[,
  predictors]), y = math.train$grade.cat, family = "multinomial",
  alpha = 1, lambda = math.logistic.lasso.cv$lambda.min, standardize = TRUE)

#### 1-r LASSO math train error-----
1 - mean(math.train$grade.cat == predict(math.logistic.lasso.best,
  as.matrix(math.train[, predictors]), type = "class"))

#### 1-r LASSO math classification-----
math.logistic.lasso <- cbind(math.test$grade.cat, predict(math.logistic.lasso.best,
  as.matrix(math.test[, predictors]), type = "class"))
colnames(math.logistic.lasso) <- c("true", "predicted")
View(math.logistic.lasso)

#### 1-r LASSO math Misclassification rate-----
1 - mean(math.test$grade.cat == predict(math.logistic.lasso.best,
  as.matrix(math.test[, predictors]), type = "class"))

### 1-r LASSO Portuguese-----
set.seed(114514)
por.logistic.lasso.cv = cv.glmnet(x = as.matrix(por.train[, predictors]),
  y = por.train$grade.cat, family = "multinomial", alpha = 1,
  lambda = exp(seq(from = -5, to = 5, length.out = 100)), standardize = TRUE)
plot(por.logistic.lasso.cv, cex.lab = 2, cex.axis = 2)
por.logistic.lasso.best = glmnet(x = as.matrix(por.train[, predictors]),
  y = por.train$grade.cat, family = "multinomial", alpha = 1,
  lambda = por.logistic.lasso.cv$lambda.min, standardize = TRUE)

#### 1-r LASSO Portuguese train error-----
1 - mean(por.train$grade.cat == predict(por.logistic.lasso.best,
  as.matrix(por.train[, predictors]), type = "class"))

#### 1-r LASSO Portuguese classification-----
por.logistic.lasso <- cbind(por.test$grade.cat, predict(por.logistic.lasso.best,
  as.matrix(por.test[, predictors]), type = "class"))
colnames(por.logistic.lasso) <- c("true", "predicted")
View(por.logistic.lasso)

#### 1-r LASSO Portuguese Misclassification rate-----
1 - mean(por.test$grade.cat == predict(por.logistic.lasso.best,
  as.matrix(por.test[, predictors]), type = "class"))

## logistic regression ridge-----

### 1-r ridge math-----
set.seed(114514)
math.logistic.ridge.cv = cv.glmnet(x = as.matrix(math.train[,

```

```

predictors]], y = math.train$grade.cat, family = "multinomial",
alpha = 0, lambda = exp(seq(from = -5, to = 5, length.out = 100)),
standardize = TRUE)
plot(math.logistic.ridge.cv, cex.lab = 2, cex.axis = 2)
math.logistic.ridge.best = glmnet(x = as.matrix(math.train[,
predictors]), y = math.train$grade.cat, family = "multinomial",
alpha = 0, lambda = math.logistic.ridge.cv$lambda.min, standardize = TRUE)

#### 1-r ridge math train error-----
1 - mean(math.train$grade.cat == predict(math.logistic.ridge.best,
as.matrix(math.train[, predictors]), type = "class"))

#### 1-r ridge math classification-----
math.logistic.ridge <- cbind(math.test$grade.cat, predict(math.logistic.ridge.best,
as.matrix(math.test[, predictors]), type = "class"))
colnames(math.logistic.ridge) <- c("true", "predicted")
View(math.logistic.ridge)

#### 1-r ridge math Misclassification rate-----
1 - mean(math.test$grade.cat == predict(math.logistic.ridge.best,
as.matrix(math.test[, predictors]), type = "class"))

### 1-r ridge Portuguese-----
set.seed(114514)
por.logistic.ridge.cv = cv.glmnet(x = as.matrix(por.train[, predictors]),
y = por.train$grade.cat, family = "multinomial", alpha = 0,
lambda = exp(seq(from = -5, to = 5, length.out = 100)), standardize = TRUE)
plot(por.logistic.ridge.cv, cex.lab = 2, cex.axis = 2)
por.logistic.ridge.best = glmnet(x = as.matrix(por.train[, predictors]),
y = por.train$grade.cat, family = "multinomial", alpha = 0,
lambda = por.logistic.ridge.cv$lambda.min, standardize = TRUE)

#### 1-r ridge por train error-----
1 - mean(por.train$grade.cat == predict(por.logistic.ridge.best,
as.matrix(por.train[, predictors]), type = "class"))

#### 1-r ridge Portuguese classification-----
por.logistic.ridge <- cbind(por.test$grade.cat, predict(por.logistic.ridge.best,
as.matrix(por.test[, predictors]), type = "class"))
colnames(por.logistic.ridge) <- c("true", "predicted")
View(por.logistic.ridge)

#### 1-r ridge Portuguese Misclassification rate-----
1 - mean(por.test$grade.cat == predict(por.logistic.ridge.best,
as.matrix(por.test[, predictors]), type = "class"))

## LDA ----- LDA
## math-----
math.lda <- lda(classification.formula, data = math.train)
math.coord.1 <- as.matrix(math.train[, predictors]) %*% math.lda$scaling[,
1]
math.coord.2 <- as.matrix(math.train[, predictors]) %*% math.lda$scaling[,

```



```

2]
ggplot(data.frame(math.coord.1, math.coord.2, math.train$grade.cat)) +
  geom_point(aes(math.coord.1, math.coord.2, color = math.train$grade.cat))

#### LDA math train error-----
1 - mean(math.train$grade.cat == predict(math.lda, newdata = math.train)$class)

#### LDA math classification-----
LDA.math.pred <- cbind(math.test$grade.cat, predict(math.lda,
  newdata = math.test)$class)
colnames(LDA.math.pred) <- c("true", "predicted")
View(LDA.math.pred)

#### LDA math Misclassification Rate-----
1 - mean(math.test$grade.cat == predict(math.lda, newdata = math.test)$class)

### LDA Portuguese-----
por.lda <- lda(classification.formula, data = por.train)
por.coord.1 <- as.matrix(por.train[, predictors]) %*% por.lda$scaling[,
1]
por.coord.2 <- as.matrix(por.train[, predictors]) %*% por.lda$scaling[,
2]
ggplot(data.frame(por.coord.1, por.coord.2, por.train$grade.cat)) +
  geom_point(aes(por.coord.1, por.coord.2, color = por.train$grade.cat))

#### LDA Portuguese train error-----
1 - mean(por.train$grade.cat == predict(por.lda, newdata = por.train)$class)

#### LDA Portuguese classification-----
LDA.por.pred <- cbind(por.test$grade.cat, predict(por.lda, newdata = por.test)$class)
colnames(LDA.por.pred) <- c("true", "predicted")
View(LDA.por.pred)

#### LDA Portuguese Misclassification Rate-----
1 - mean(por.test$grade.cat == predict(por.lda, newdata = por.test)$class)

# Regression -----

## OLS ----- OLS
## math-----
math.ols <- lm(regression.formula, data = por.train)

### OLS math train error-----
mean((math.train$grade.con - predict(math.ols, newdata = math.train[,
predictors]))^2)

#### OLS math prediction-----
math.ols.pred <- (cbind(math.test$grade.con, predict(math.ols,
  math.test)))
colnames(math.ols.pred) <- c("true", "predicted")
View(math.ols.pred)

```

```

#### OLS math Mean Squared Error-----
mean((math.test$grade.con - predict(math.ols, newdata = math.test[,
  predictors]))^2)

### OLS Portuguese-----
por.ols <- lm(regression.formula, data = por.train)

#### OLS Portuguese train error-----
mean((por.train$grade.con - predict(por.ols, newdata = por.train[,
  predictors]))^2)

#### OLS Portuguese prediction-----
por.ols.pred <- (cbind(por.test$grade.con, predict(por.ols, por.test)))
colnames(por.ols.pred) <- c("true", "predicted")
View(por.ols.pred)

#### OLS Portuguese Mean Squared Error-----
mean((por.test$grade.con - predict(por.ols, newdata = por.test[,
  predictors]))^2)

## Ridge ----- ridge
## math-----
set.seed(114514)
math.ridge.cv <- cv.glmnet(as.matrix(math.train[, predictors]),
  math.train$grade.con, alpha = 0, lambda = exp(seq(-5, 5,
    by = 0.1)), standardize = TRUE)
plot(math.ridge.cv)
math.ridge.best <- glmnet(math.train[, predictors], math.train$grade.con,
  alpha = 0, lambda = math.ridge.cv$lambda.min, standardize = TRUE)

#### ridge math error rate-----
mean((math.train$grade.con - predict(math.ridge.best, as.matrix(math.train[,
  predictors]))))^2)

#### ridge math prediction-----
math.ridge.pred <- (cbind(math.test$grade.con, predict(math.ridge.best,
  as.matrix(math.test[, predictors]))))
colnames(math.lasso.pred) <- c("true", "predicted")
View(math.ridge.pred)

#### ridge math Mean Squared Error-----
mean((math.test$grade.con - predict(math.ridge.best, as.matrix(math.test[,
  predictors]))))^2)

### ridge Portuguese-----

set.seed(114514)
por.ridge.cv <- cv.glmnet(as.matrix(por.train[, predictors]),
  por.train$grade.con, alpha = 0, lambda = exp(seq(-5, 5, by = 0.1)),
  standardize = TRUE)
plot(por.ridge.cv)
por.ridge.best <- glmnet(por.train[, predictors], por.train$grade.con,
  alpha = 0, lambda = por.ridge.cv$lambda.min, standardize = TRUE)

```

```

#### ridge Portuguese train error-----
mean((por.train$grade.con - predict(por.ridge.best, as.matrix(por.train[,
predictors]))))^2)

#### ridge Portuguese prediction-----
por.ridge.pred <- (cbind(por.test$grade.con, predict(por.ridge.best,
as.matrix(por.test[, predictors]))))
colnames(math.lasso.pred) <- c("true", "predicted")
View(por.ridge.pred)

#### ridge Portuguese Mean Squared Error-----
mean((por.test$grade.con - predict(por.ridge.best, as.matrix(por.test[,
predictors]))))^2)

## LASSO ----- LASSO
## math-----
set.seed(114514)
math.lasso.cv <- cv.glmnet(as.matrix(math.train[, predictors]),
math.train$grade.con, alpha = 1, lambda = exp(seq(-5, 5,
by = 0.1)), standardize = TRUE)
plot(math.lasso.cv)
math.lasso.best <- glmnet(math.train[, predictors], math.train$grade.con,
alpha = 1, lambda = math.lasso.cv$lambda.min, standardize = TRUE)

#### LASSO math train error-----
mean((math.train$grade.con - predict(math.lasso.best, as.matrix(math.train[,
predictors]))))^2)

#### LASSO math prediction-----
math.lasso.pred <- (cbind(math.test$grade.con, predict(math.lasso.best,
as.matrix(math.test[, predictors]))))
colnames(math.lasso.pred) <- c("true", "predicted")
View(math.lasso.pred)

#### LASSO math Mean Squared Error-----
mean((math.test$grade.con - predict(math.lasso.best, as.matrix(math.test[,
predictors]))))^2)

#### Math Important variable-----
coef(math.lasso.best)

### LASSO Portuguese-----
set.seed(114514)
por.lasso.cv <- cv.glmnet(as.matrix(por.train[, predictors]),
por.train$grade.con, alpha = 1, lambda = exp(seq(-5, 5, by = 0.1)),
standardize = TRUE)
plot(por.lasso.cv)
por.lasso.best <- glmnet(por.train[, predictors], por.train$grade.con,
alpha = 1, lambda = por.lasso.cv$lambda.min, standardize = TRUE)

#### LASSO Portuguese train error-----
mean((por.train$grade.con - predict(por.lasso.best, as.matrix(por.train[,
predictors]))))^2)

```

```

#### LASSO por prediction-----
por.lasso.pred <- (cbind(por.test$grade.con, predict(por.lasso.best,
  as.matrix(por.test[, predictors]))))
colnames(math.lasso.pred) <- c("true", "predicted")
View(por.lasso.pred)

#### LASSO Portuguese Mean Squared Error-----
mean((por.test$grade.con - predict(por.lasso.best, as.matrix(por.test[,
  predictors])))^2)

#### Portuguese important variable-----
coef(por.lasso.best)

## SVM regression -----

set.seed(114514)
cost.list = exp(seq(from = -8, to = 0, length.out = 100))
math.svm.cv.val.error.reg = svm.cv.cost.reg(q3.train = math.train,
  form = regression.formula, k = 10, response_variable = "grade.con",
  cost.list = cost.list)
ggplot() + geom_line(aes(x = log(cost.list), y = math.svm.cv.val.error.reg),
  linewidth = 1.5) + theme(axis.text.x = element_text(size = 20,
  angle = 0, vjust = 0.3), axis.text.y = element_text(size = 20),
  plot.title = element_text(size = 20), axis.title.x = element_text(size = 20),
  axis.title.y = element_text(size = 20), legend.position = "none")

math.svm.fit.reg = svm(regression.formula, data = math.train,
  cost = cost.list[which.min(math.svm.cv.val.error.reg)], kernel = "linear")
math.svm.error.train.reg = mean((predict(math.svm.fit.reg) -
  math.train$grade.con)^2)
math.svm.error.test.reg = mean((predict(math.svm.fit.reg, math.test) -
  math.test$grade.con)^2)

set.seed(114514)
cost.list = exp(seq(from = -8, to = 0, length.out = 100))
por.svm.cv.val.error.reg = svm.cv.cost.reg(q3.train = por.train,
  form = regression.formula, k = 10, response_variable = "grade.con",
  cost.list = cost.list)
ggplot() + geom_line(aes(x = log(cost.list), y = por.svm.cv.val.error.reg),
  linewidth = 1.5) + theme(axis.text.x = element_text(size = 20,
  angle = 0, vjust = 0.3), axis.text.y = element_text(size = 20),
  plot.title = element_text(size = 20), axis.title.x = element_text(size = 20),
  axis.title.y = element_text(size = 20), legend.position = "none")

por.svm.fit.reg = svm(regression.formula, data = por.train, cost = cost.list[which.min(por.svm.cv.val.e
  kernel = "linear")
por.svm.error.train.reg = mean((predict(por.svm.fit.reg) - por.train$grade.con)^2)
por.svm.error.test.reg = mean((predict(por.svm.fit.reg, por.test) -
  por.test$grade.con)^2)

```