Homework 2

Zhezheng Jin

Contents

```
library(tidyverse)
library(caret)
library(tidymodels)
library(earth)
library(splines)
library(mgcv)
library(pdp)
library(bayesQR)
```

```
# Data Import
data = read_csv("College.csv") %>%
janitor::clean_names() %>%
select(-college) %>%
relocate(outstate)
```

```
## Rows: 565 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): College
## dbl (17): Apps, Accept, Enroll, Top10perc, Top25perc, F.Undergrad, P.Undergr...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# data partition
set.seed(80)
indexTrain <- createDataPartition(y = data$outstate, p = 0.8, list = FALSE)
trainData <- data[indexTrain, ]
testData <- data[-indexTrain, ]
```

The "College" dataset contains 17 columns and 565 observations after omitting the `college` variable. Then we partition the dataset into two parts: training data (80%) and test data (20%), where the training data and test data contains 453 and 112 rows, respectively.

```
# matrix of predictors
x <- model.matrix(outstate~.,trainData)[,-1]
x2 <- model.matrix(outstate~.,testData)[,-1]

# vector of response
y <- trainData$outstate
y2 <- testData$outstate

# 10-fold cv on best
ctrl1 <- trainControl(method = "cv", number = 10)
```

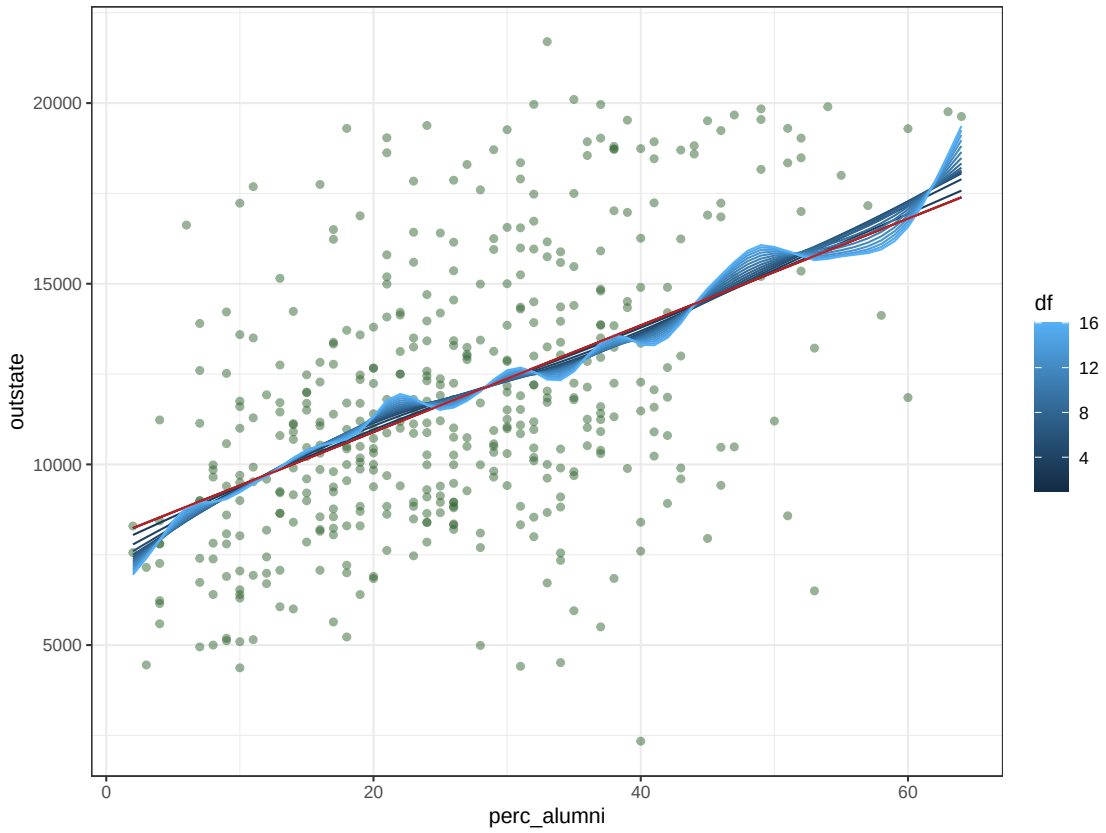
(a) smoothing spline models

```
# the range of perc_alumni is [2, 64]
perc_alumni.grid <- seq(from =2, to =64,by =1)
```

```
# write a function to apply multiple df's to smoothing spline
ss.data.func <- function(trainData, perc_alumni.grid, df_seq) {
  ss.list <- lapply(df_seq, function(df) {
    fit.ss <- smooth.spline(trainData$perc_alumni, trainData$outstate, df = df)
    pred.ss <- predict(fit.ss, x = perc_alumni.grid)
    pred.ss.df <- data.frame(pred = pred.ss$y,
                             perc.alumni = perc_alumni.grid, df = df)
    return(pred.ss.df)
  })
  ss.data <- do.call(rbind, ss.list)
  return(ss.data)
}

p <- ggplot(data = trainData, aes(x = perc_alumni, y = outstate)) +
  geom_point(color = rgb(.2, .4, .2, .5)) +
  geom_line(aes(x = perc.alumni, y = pred, group = df, color = df),
            data = ss.data.func(trainData, perc_alumni.grid, 1:16))

# degree of freedom obtained by generalized cross-validation
fit.ss <- smooth.spline(trainData$perc_alumni, trainData$outstate)
fit.ss$df
```

```
## [1] 2.000218
```

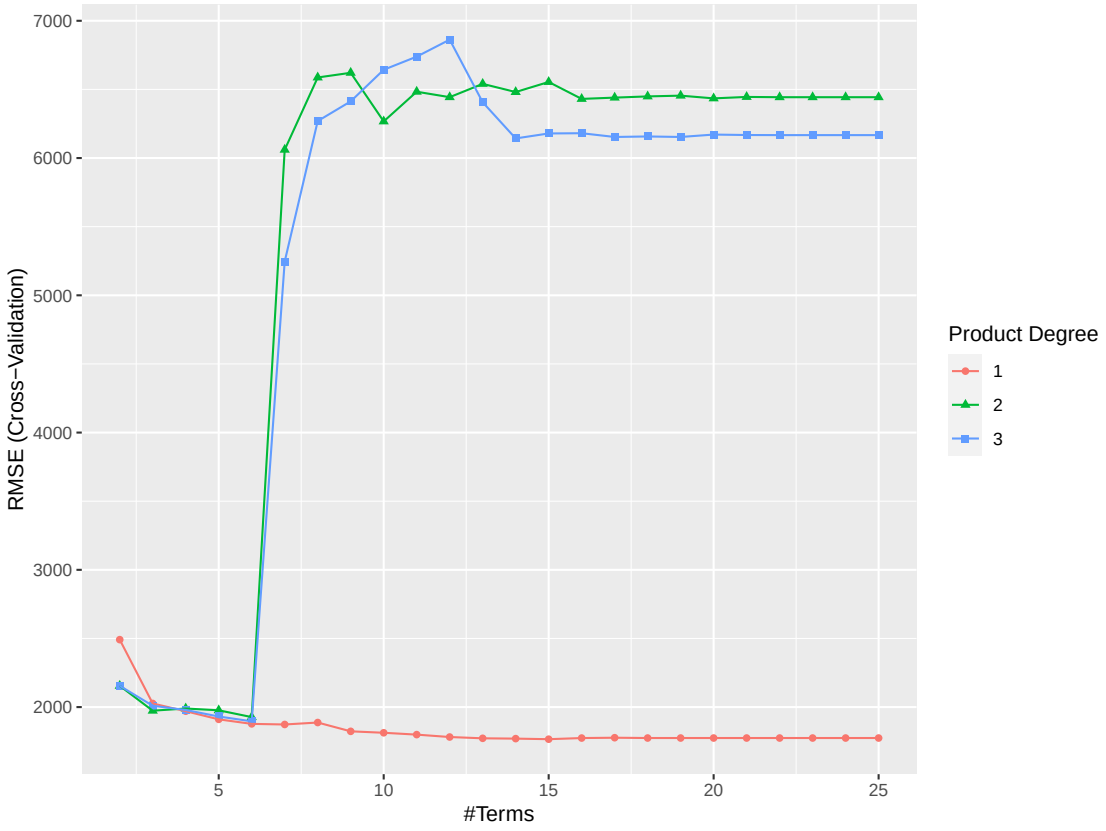
```
pred.ss <- predict(fit.ss, x = perc_alumni.grid)
pred.ss.df <- data.frame(pred = pred.ss$y,
                         perc_alumni = perc_alumni.grid)

# plot the resulting fits
p + geom_line(aes(x = perc_alumni, y = pred),
              data = pred.ss.df,color = rgb(.8, .1, .1, 1)) +
  theme_bw()
```



We can see from the plot that the curve is asymptotically to a line with degree of freedom decreasing. The appropriate degree of freedom for the model I choose is the df obtained by generalized cross-validation: 2.0002176, and we can see the resulting fits curve in red is pretty smooth, which means the model is less sensitive to noise in the data. The out-of-state tuition `outstate` increases by the percentage of alumni who donate `perc.alumni`. Therefore, the criteria for determining the best choice of degree of freedom would include minimization of the generalized cross-validation score.

(b) MARS

```
mars_grid <- expand.grid(degree = 1:3,
                         nprune = 2:25)

set.seed(80)
mars.fit <- train(x, y,
                  method = "earth",
                  tuneGrid = mars_grid,
                  trControl = ctrl1)

ggplot(mars.fit)
```




```
mars.fit$bestTune
```

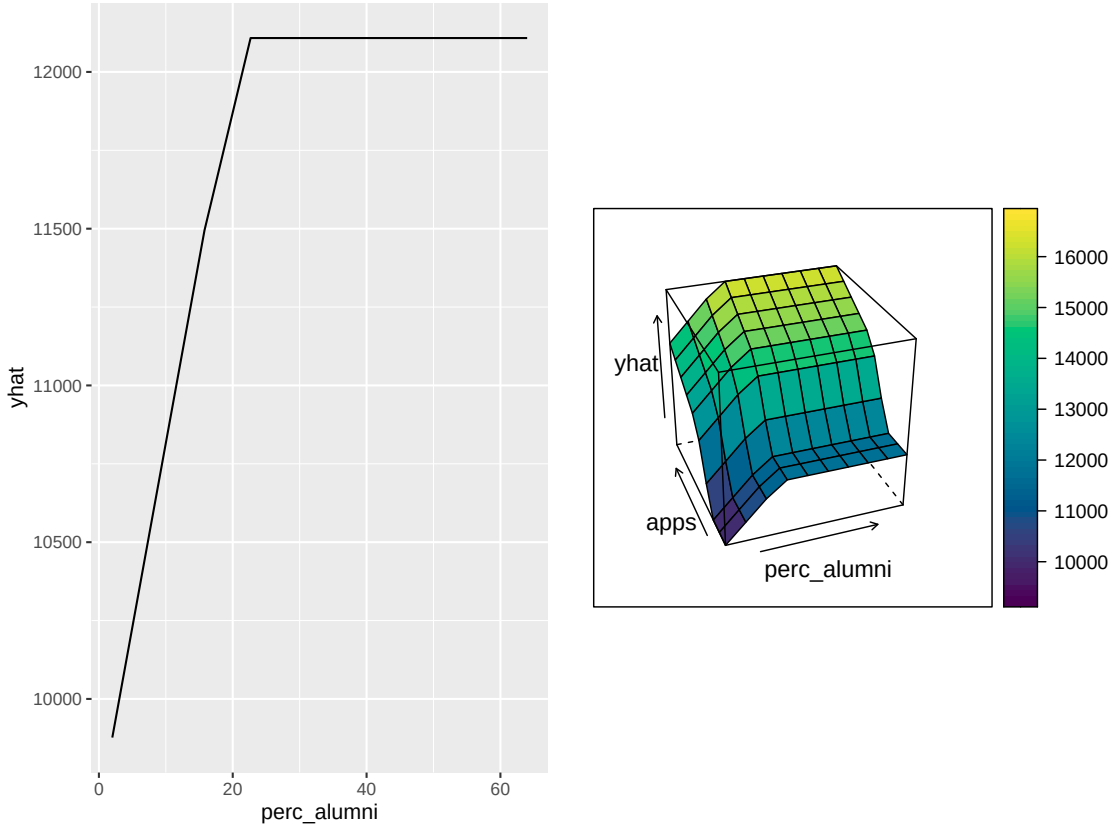
```
##    nprune degree
## 14     15      1
```

```
coef(mars.fit$finalModel)
```

```
##          (Intercept)     h(expend-15687)     h(grad_rate-83)  h(4310-room_board)
##         9781.6929305          -0.7056607       -1310.0711678          -1.2583568
##   h(21-perc_alumni) h(f_undergrad-1411) h(1411-f_undergrad)        h(apps-7033)
##         -117.4413225          -0.3945817          -1.4746215          -0.6538507
##    h(1250-personal)       h(910-enroll)      h(terminal-75)     h(grad_rate-82)
##           0.9427366           4.9140547          43.2078705        1234.1439723
##      h(expend-6875)      h(2279-accept)        h(apps-3624)
##           0.6789122          -1.7077408           0.8767452
```

```
# Present the partial dependence plot of an arbitrary predictor
p1 <- pdp::partial(mars.fit, pred.var = c("perc_alumni"), grid.resolution = 10) %>%
  autoplot()

p2 <- pdp::partial(mars.fit, pred.var =
                     c("perc_alumni", "apps"),
                   grid.resolution = 10) %>%
  pdp::plotPartial(levelplot = FALSE, zlab = "yhat", drape = TRUE,
                   screen = list(z = 20, x = -60))
```

```
gridExtra::grid.arrange(p1, p2, ncol = 2)
```




```
# test error
mars.pred <- predict(mars.fit, newdata = x2)
mars.test.error <- mean((mars.pred - y2)^2)
mars.test.error
```

```
## [1] 3124795
```

The regression function is as follow:

$$outstate = 9781.69 - 0.706 \cdot h(expend - 15687) - 1310.07 \cdot h(grad\_rate - 83) - 1.258 \cdot h(room\_board - 4310)$$
$$-117.44 \cdot h(perc\_alumni - 21) - 0.395 \cdot h(f\_undergrad - 1411) - 1.475 \cdot h(1411 - f\_undergrad)$$
$$-0.654 \cdot h(apps - 7033) + 0.943 \cdot h(personal - 1250) + 4.914 \cdot h(enroll - 910) + 43.21 \cdot h(terminal - 75)$$
$$+1234.14 \cdot h(grad\_rate - 82) + 0.679 \cdot h(expend - 6875) - 1.708 \cdot h(2279 - accept) + 0.877 \cdot h(apps - 3624)$$

The partial dependence plot shows a linear increasing of `outstate` as `perc_alumni` increases with a cut point at about 23 % of alumni who donate, holding other variables constant. Also, the plot of `perc_alumni` and `apps` is also shown to visualize their impact on the outcome `outstate`.

The test error is 3.1247948×10^6.

(c) GAM

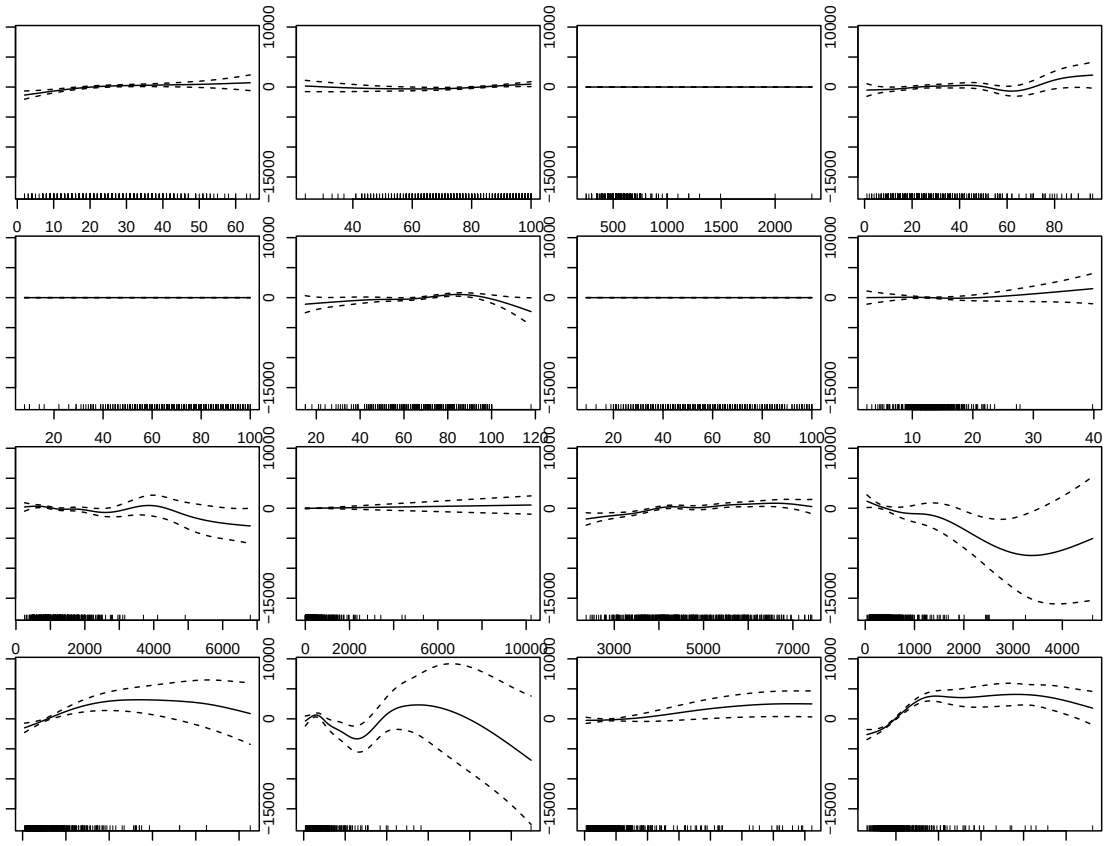
```
# use all the predictors
set.seed(80)
gam.fit <- train(x, y,
                 method = "gam",
                 tuneGrid = data.frame(method = "GCV.Cp", select = TRUE),
                 trControl = ctrl1)

gam.fit$bestTune
```

```
##   select method
## 1   TRUE GCV.Cp
```

```
gam.fit$finalModel
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## .outcome ~ s(perc_alumni) + s(terminal) + s(books) + s(top10perc) +
##     s(ph_d) + s(grad_rate) + s(top25perc) + s(s_f_ratio) + s(personal) +
##     s(p_undergrad) + s(room_board) + s(enroll) + s(accept) +
##     s(f_undergrad) + s(apps) + s(expend)
##
## Estimated degrees of freedom:
## 2.787 1.887 0.000 6.140 0.000 4.601 0.000
## 2.246 5.463 0.354 5.525 3.812 2.747 6.688
## 1.583 5.340  total = 50.17
##
## GCV score: 2757010
```

```
# Plot the results
par(mar = c(1, 1, 1, 1), mfrow=c(4,4))
plot(gam.fit$finalModel)
```



```
# test error
gam.pred <- predict(gam.fit, newdata = x2)
gam.test.error <- mean((gam.pred - y2)^2)
gam.test.error
```

```
## [1] 3673235
```

The GAM model includes all the predictors. There is no bivariate function in the model, which means no interaction between predictors.The plot shows the 16 predictors (from left to the right, from top to the bottom) perc_alumni, terminal, books, ph_d, top10perc, grad_rate, top25perc, s_f_ratio, personal, p_undergrad, enroll, room_board, accept, f_undergrad, apps, expend. Among these predictors, it seems like there are 8 nonlinear terms: `room_board`, `grad_rate`,`top25perc`, `p_undergrad`,`enroll`, `f_undergrad`, `apps`, `expend`.

The test error is 3.6732353×10^6.

(d) MARS model & linear model comparison

```
# Fit the linear model
set.seed(80)
lm.fit <- train(x, y, method = "lm", trControl = ctrl1)

# Test error
lm.pred <- predict(lm.fit, newdata = x2)
```

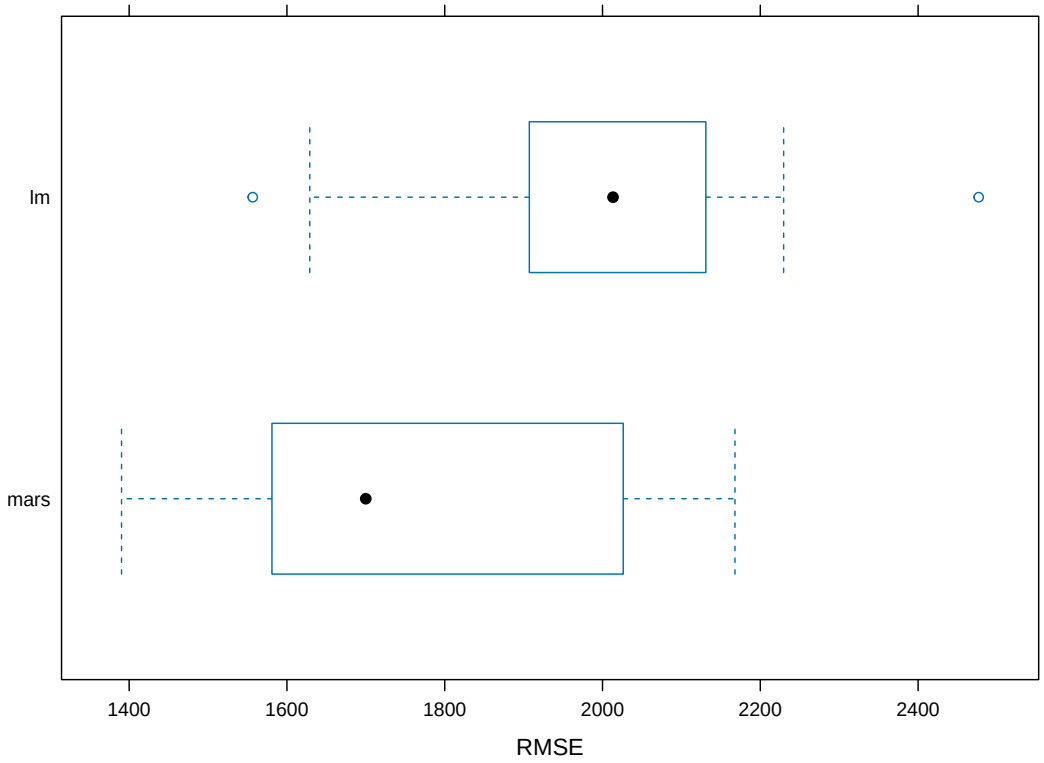
```r
lm.test.error <- mean((lm.pred - y2)^2)
lm.test.error
```

```
## [1] 3959583
```

```r
# resamples
resamp <- resamples(list(lm = lm.fit,
                         mars = mars.fit))
summary(resamp)
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: lm, mars
## Number of resamples: 10
##
## MAE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm   1248.698 1553.354 1604.223 1579.881 1670.531 1775.242    0
## mars 1136.547 1258.262 1344.897 1354.885 1406.098 1665.248    0
##
## RMSE
##          Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## lm   1556.623 1911.264 2013.309 1990.244 2103.822 2476.792    0
## mars 1390.360 1599.039 1699.909 1766.085 1981.679 2167.859    0
##
## Rsquared
##           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## lm   0.5826814 0.6985075 0.7357962 0.7236187 0.7661080 0.8142127    0
## mars 0.6887113 0.7392714 0.8161305 0.7821655 0.8197901 0.8292302    0
```

```r
# RMSE box-plot between models
bwplot(resamp, metric = "RMSE")
```



Since MARS model has a obvious lower mean RMSE, we prefer to use MARS model over a linear model when predicting the out-of-state tuition in this data example.

Usually, MARS can outperform linear models, especially when the relationships between predictor variables and the response variable are non-linear. MARS does this by creating piecewise linear models, which can be more flexible than a linear model.

However, whether MARS is a better approach compared to a linear model depends on the specific application and the nature of the data. In some cases, the relationships between predictors and the response variable may be linear, in which a linear model would be better.

In summary, the choice of modeling technique should be based on the nature of the data and the specific research questions. It may also be useful to compare the performance of different modeling techniques using cross-validation to determine which approach is the best for a specific application.