hw5

Question 1 Step 1. Import data, imputation for missing values, and set categorical data

```r
dat <- read.csv("C:/Users/linzh/Desktop/ECON 626/hw5/aug_train.csv")

#categorical variable: gender
unique(dat$gender)
```

```
## [1] "Male"   ""       "Female" "Other"
```

```r
mapping <- c("Male" = 0, "Female" = 1, "Other" = 2)
dat$gender <- mapping[dat$gender]
dat$gender[is.na(dat$gender)] <- 2

#categorical variable: relevent_experience
unique(dat$relevent_experience)
```

```
## [1] "Has relevent experience" "No relevent experience"
```

```r
mapping <- c("Has relevent experience" = 0, "No relevent experience" = 1)
dat$relevent_experience <- mapping[dat$relevent_experience]

#categorical variable: enrolled_university
unique(dat$enrolled_university)
```

```
## [1] "no_enrollment"    "Full time course" ""                 "Part time course"
```

```r
mapping <- c("Full time course" = 0, "Part time course" = 1, "no_enrollment" = 2)
dat$enrolled_university <- mapping[dat$enrolled_university]
dat$enrolled_university[is.na(dat$enrolled_university)] <- 9

#categorical variable: education_level
unique(dat$education_level)
```

```
## [1] "Graduate"       "Masters"        "High School"    ""
## [5] "Phd"            "Primary School"
```

```r
mapping <- c("Graduate" = 0, "Masters" = 1, "High School" = 2, "Phd" = 3, "Primary School" = 4)
dat$education_level <- mapping[dat$education_level]
dat$education_level[is.na(dat$education_level)] <- 9

#categorical variable: major_discipline
unique(dat$major_discipline)
```

```
## [1] "STEM"            "Business Degree" ""                "Arts"
## [5] "Humanities"      "No Major"        "Other"
```

```
mapping <- c("STEM" = 0, "Business Degree" = 1, "Arts" = 2, "Humanities" = 3, "No Major" = 4,
             "Other" = 5)
dat$major_discipline <- mapping[dat$major_discipline]
dat$major_discipline[is.na(dat$major_discipline)] <- 5

#categorical variable: experience
unique(dat$experience)
```

```
##  [1] ">20" "15"  "5"   "<1"  "11"  "13"  "7"   "17"  "2"   "16"  "1"   "4"
## [13] "10"  "14"  "18"  "19"  "12"  "3"   "6"   "9"   "8"   "20"  ""
```

```
dat$experience <- as.factor(dat$experience)

#categorical variable: company_size
unique(dat$company_size)
```

```
## [1] ""          "50-99"     "<10"       "10000+"    "5000-9999" "1000-4999"
## [7] "10/49"     "100-500"   "500-999"
```

```
mapping <- c("<10" = 0, "10/49" = 1, "50-99" = 2, "100-500" = 3, "500-999" = 4, "1000-4999" = 5,
             "5000-9999" = 6, "10000+" = 7)
dat$company_size <- mapping[dat$company_size]
dat$company_size[is.na(dat$company_size)] <- 9

#categorical variable: company_type
unique(dat$company_type)
```

```
## [1] ""                    "Pvt Ltd"             "Funded Startup"
## [4] "Early Stage Startup" "Other"               "Public Sector"
## [7] "NGO"
```

```
mapping <- c("Pvt Ltd" = 0, "Funded Startup" = 1, "Early Stage Startup" = 2, "Public Sector" = 3,
             "NGO" = 4, "Other" = 5)
dat$company_type <- mapping[dat$company_type]
dat$company_type[is.na(dat$company_type)] <- 5

#categorical variable: last_new_job
unique(dat$last_new_job)
```

```
## [1] "1"     ">4"    "never" "4"     "3"     "2"     ""
```

```
mapping <- c("1" = 0, "2" = 1, "3" = 2, "4" = 3, ">4" = 4, "never" = 5)
dat$last_new_job <- mapping[dat$last_new_job]
dat$last_new_job[is.na(dat$last_new_job)] <- 9

#categorical variable: training_hours
dat$training_hours <- cut(dat$training_hours, breaks=c(0, 50, 100, 150, 200, 250,
                        300), labels=c(0, 1, 2, 3, 4, 5))
mapping <- c("0" = 0, "1" = 1, "2" = 2, "3" = 3, "4" = 4, "5" = 5, "NA" = 6)
dat$training_hours <- mapping[dat$training_hours]
dat$training_hours[is.na(dat$training_hours)] <- 6
```

Step 2. Separate train and test datasets

```
set.seed(9)
num_obs <- nrow(dat)
dat_index <-  sample(num_obs, size = trunc(0.2 * num_obs))# randomly select 2/10 data
train <- dat[dat_index, ]# Training sample
```

Step 3. Fit the SVM model and report train data accuracy

```
set.seed(1)
tune_out <- tune(svm, target ~ . - enrollee_id - city, data = train, kernel = "radial",
                ranges = list(cost = c(0.1,1,10,100,1000), gamma = c(0.1,0.5,1,2,3,4)))
bestmod <- tune_out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = target ~ . - enrollee_id - city,
##     data = train, ranges = list(cost = c(0.1, 1, 10, 100, 1000),
##         gamma = c(0.1, 0.5, 1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##     epsilon:  0.1
##
##
## Number of Support Vectors:  2923
```

```
train_prediction <- predict(bestmod, newdata = train)
train_pred <- ifelse(train_prediction > 0.5, 1, 0)

# confusion matrix and accuracy
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
comparison <- table(train_pred, train$target)
confusionMatrix(comparison)
```

```
## Confusion Matrix and Statistics
##
##
## train_pred    0    1
##          0 2788  290
##          1   72  681
##
##                Accuracy : 0.9055
##                  95% CI : (0.8958, 0.9146)
##     No Information Rate : 0.7465
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7303
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9748
##             Specificity : 0.7013
##          Pos Pred Value : 0.9058
##          Neg Pred Value : 0.9044
##              Prevalence : 0.7465
##          Detection Rate : 0.7277
##    Detection Prevalence : 0.8034
##       Balanced Accuracy : 0.8381
##
##        'Positive' Class : 0
##
```

Step 4. Fit the test data into model and report test data accuracy

```
test_prediction <- predict(bestmod, newdata = dat)
test_pred <- ifelse(test_prediction > 0.5, 1, 0)

# confusion matrix and accuracy
library(caret)
comparison <- table(test_pred, dat$target)
confusionMatrix(comparison)
```

```
## Confusion Matrix and Statistics
##
##
## test_pred     0     1
##         0 13359  2933
##         1  1022  1844
##
##                Accuracy : 0.7936
##                  95% CI : (0.7878, 0.7993)
##     No Information Rate : 0.7507
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3635
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
##             Sensitivity : 0.9289
##             Specificity : 0.3860
##          Pos Pred Value : 0.8200
##          Neg Pred Value : 0.6434
##              Prevalence : 0.7507
##          Detection Rate : 0.6973
##    Detection Prevalence : 0.8504
##       Balanced Accuracy : 0.6575
##
##        'Positive' Class : 0
##
```

Question 2 a) Draw the ROC curve. How do the concepts “sensitivity” and “specificity” relate to the ROC curve? b) Calculate the AUC (area under ROC). What is the interpretation of the AUC concept?

```r
# define a function to draw the roc plot.
rocplot <- function(pred, truth, ...){
   predob <- prediction(pred, truth)
   perf <- performance(predob, "tpr", "fpr")
   plot(perf,...)}

# use the parameters from the best tuned svm model.
svmfit_opt <- svm(target ~ . - enrollee_id - city, data = train, kernel = "radial",
                 gamma = 0.5, cost = 1, decision.values = TRUE)

svmfit_flex <- svm(target ~ . - enrollee_id - city, data = train, kernel = "radial",
                  gamma = 50, cost = 1, decision.values = TRUE)

par(mfrow = c(1,2))

# Plot optimal parameter model's performance on training data
fitted_opt_train <- attributes(predict(svmfit_opt, train,
                                       decision.values = TRUE))$decision.values

# draw the ROC curve of optimal parameter model's
rocplot(fitted_opt_train, train$target, main = "Training Data", col = 'blue') +
  abline(coef = c(0,1), lty = 3, col = 'red')
```

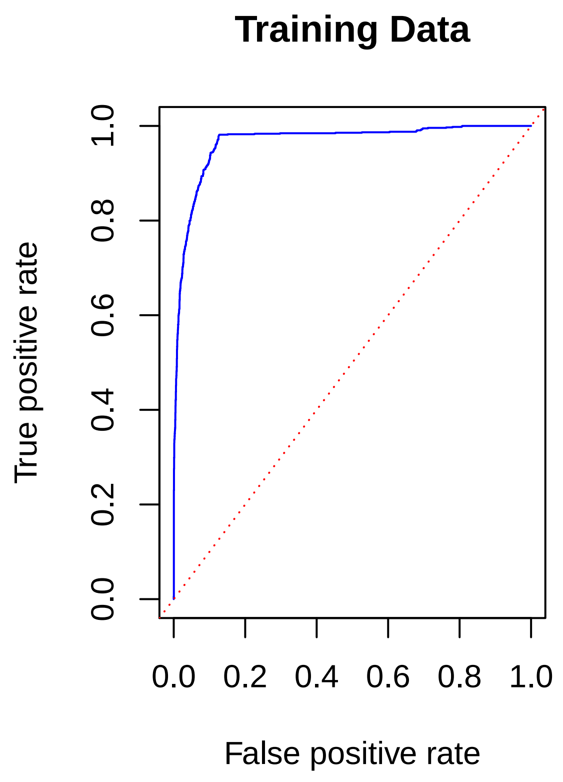
```
## integer(0)
```

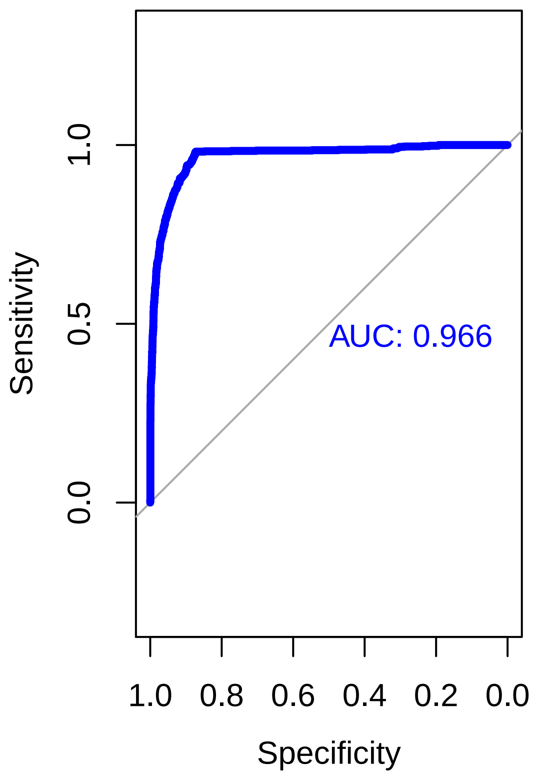
```r
# calculate the AUC of optimal parameter model's
svmROC <- roc(train$target ~ fitted_opt_train, plot = TRUE, print.auc=TRUE,col="blue",lwd=4)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(response, predictors[, 1], ...): Deprecated use a matrix
## as predictor. Unexpected results may be produced, please pass a numeric vector.
```

```
## Setting direction: controls < cases
```





```
# Plot flexible parameter model's performance on training data
fitted_flex_train <- attributes(predict(svmfit_flex, train,
                                         decision.values = TRUE))$decision.values
# draw the ROC curve of flexible parameter model's
rocplot(fitted_flex_train, train$target, col = "orange") + abline(coef=c(0,1),lty=3,col='red')
```

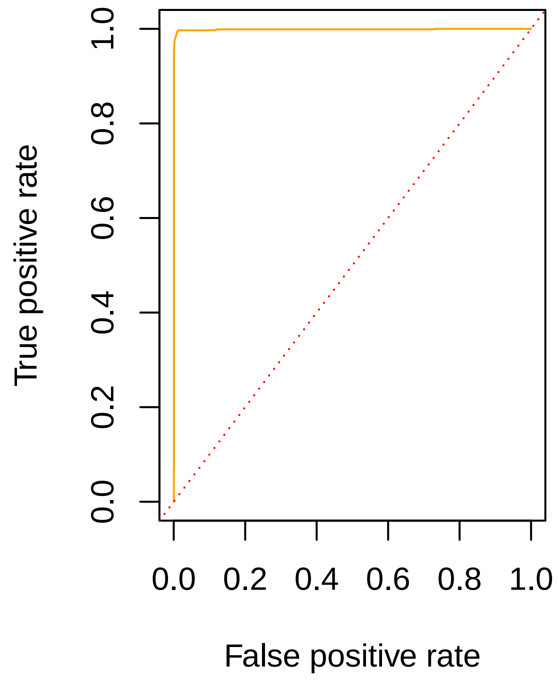
```
## integer(0)
```

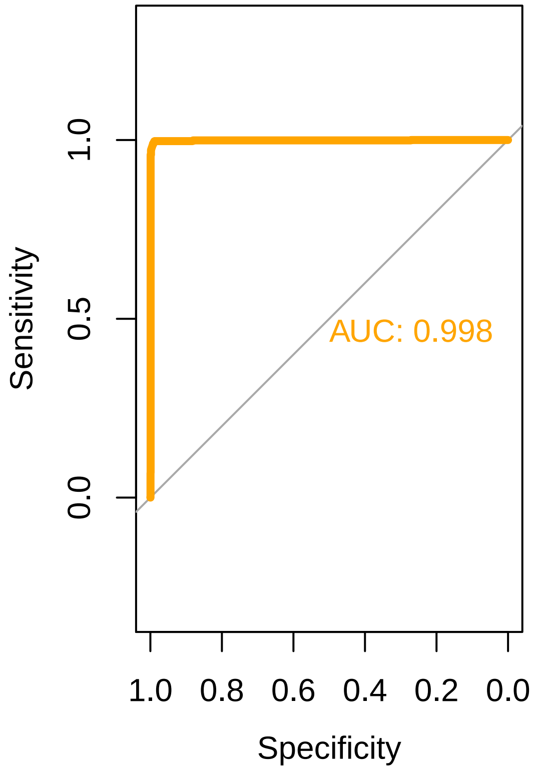
```
# calculate the AUC of flexible parameter model's
svmROC <- roc(train$target ~ fitted_flex_train, plot = TRUE, print.auc=TRUE,col="orange",lwd=4)
```

```
## Setting levels: control = 0, case = 1
```

```
## Warning in roc.default(response, predictors[, 1], ...): Deprecated use a matrix
## as predictor. Unexpected results may be produced, please pass a numeric vector.
```

```
## Setting direction: controls < cases
```





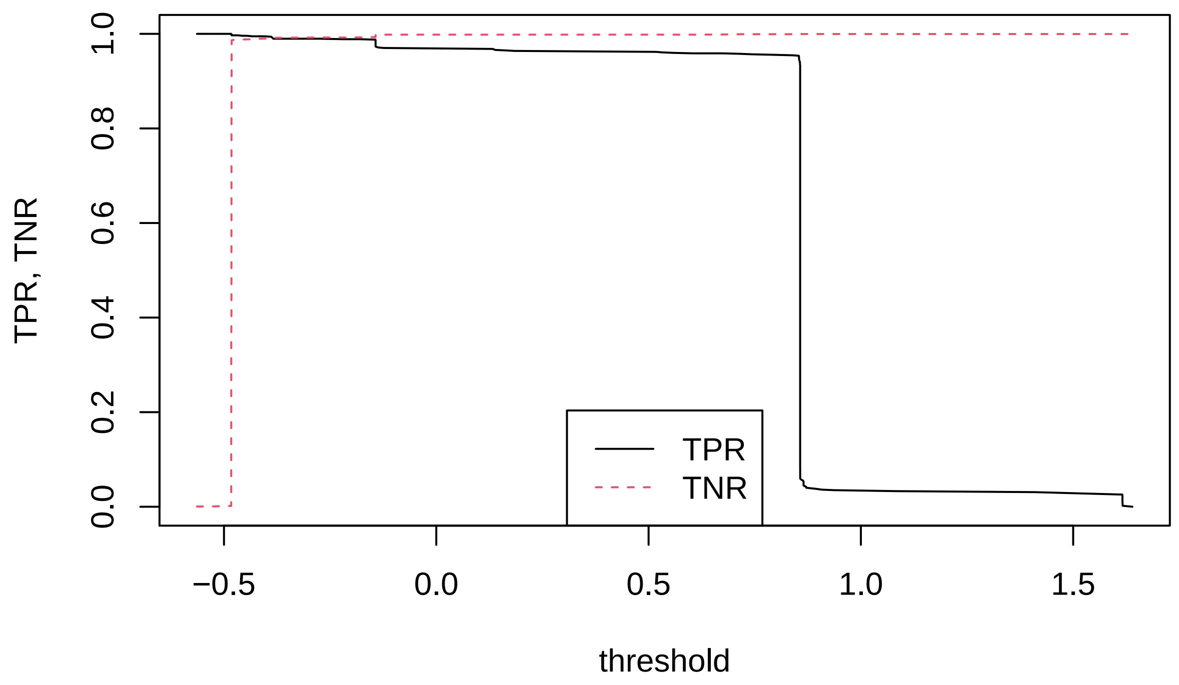
ROC curves show the trade-off relationship between sensitivity and specificity in the model.

The AUC measures the ability of a classifier to distinguish between the positive and negative classes. A higher AUC represents a model has a better ability to distinguish between classes.

By increasing gamma from 0.5 to 50, a more flexible fit is produced and and generates an improvements in accuracy: AUC increases from 0.966 to 0.998.

c) Draw a figure that illustrates how the importance of individual variables changes as the prediction threshold changes.

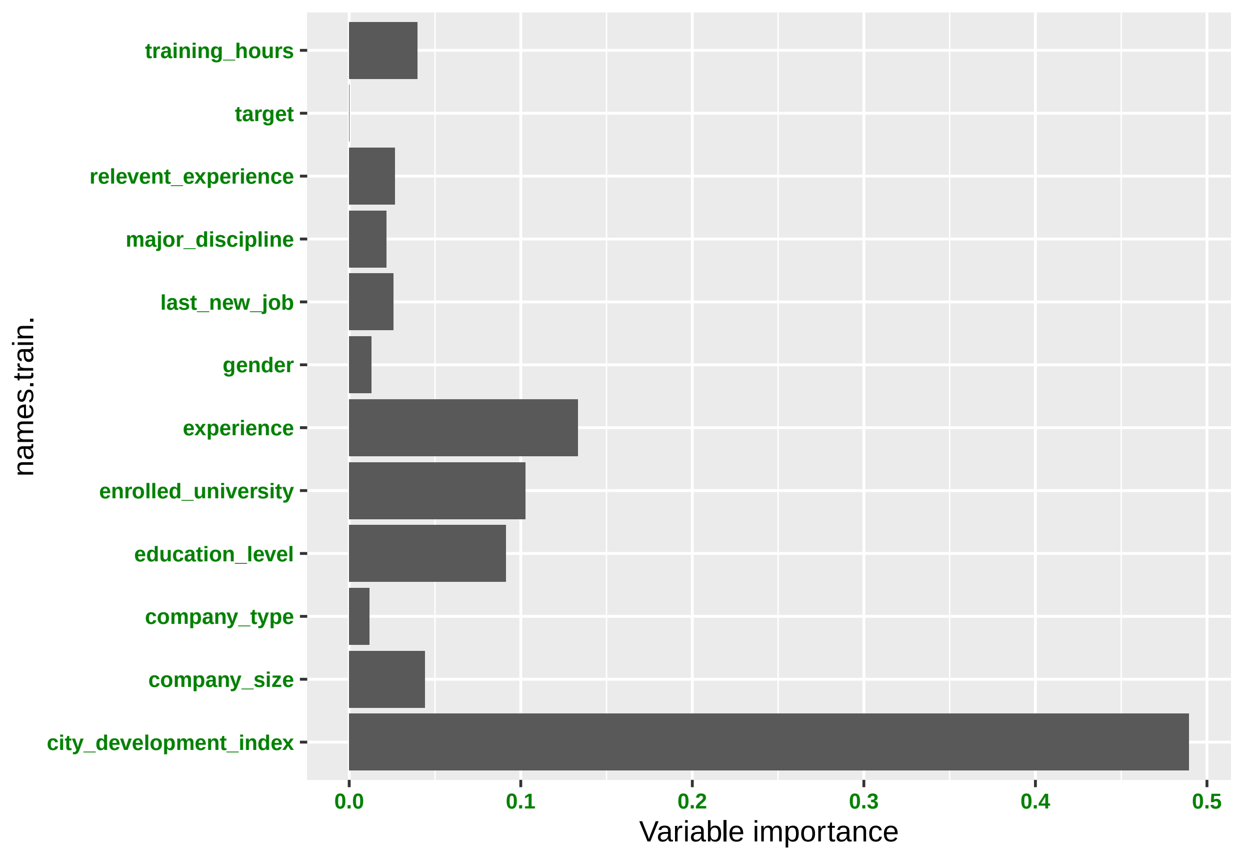
```
matplot(data.frame(svmROC$sensitivities, svmROC$specificities), x = svmROC$thresholds, type='l',
        xlab = 'threshold', ylab = 'TPR, TNR')
legend('bottom', legend = c('TPR', 'TNR'), lty = 1:2, col = 1:2)
```



d) Draw a figure that illustrates variable importance for the model

```
train <- train[, 3:14]
M <- fit(target~., data = train, model="svm", kernel = "rbfdot", kpar="automatic", C = 1,
         target = 'class')
I <- Importance(M, train, method = "1D-SA", measure = "gradient") #get variable importance

# draw a bar plot to illustrate variable importance
importances <- data.frame(names(train) ,I$imp)
importances %>%
  ggplot(aes(x = names.train., y = I.imp, main="Variable importance")) +
  geom_bar(stat = "identity") + coord_flip() + scale_y_continuous(name="Variable importance") +
  theme(axis.text.x = element_text(face="bold", color="#008000", size=8, angle=0),
          axis.text.y = element_text(face="bold", color="#008000",size=8, angle=0))
```



names.train.
training_hours
target
relevent_experience
major_discipline
last_new_job
gender
experience
enrolled_university
education_level
company_type
company_size
city_development_index
0.0
0.1
0.2
0.3
0.4
0.5
Variable importance