PREDICTION COMPETITION 5

Loading data and libraries:

```r
library(readr)
dat <- read.csv("Desktop/aug_train (1).csv", header = TRUE)
dat2<- dat
library(e1071)
library(dplyr)
```

Cleaning the data, making factors, regulating factors and creating a data frame. Dealing with missing values by reducing the imbalance in the data and making the missing values as NA

```r
### data cleaning :
dat$city = as.factor(dat$city)
dat$city_development_index = as.factor(dat$city_development_index)
dat$gender = as.factor(dat$gender)
dat$relevent_experience = as.factor(dat$relevent_experience)
dat$enrolled_university = as.factor(dat$enrolled_university)
dat$education_level = as.factor(dat$education_level)
dat$major_discipline = as.factor(dat$major_discipline)
dat$experience= as.factor(dat$experience)
dat$company_size = as.factor(dat$company_size)
dat$company_type = as.factor(dat$company_type)
dat$last_new_job = as.factor(dat$last_new_job)
dat$city<- factor(dat$city,levels = c("city_103","city_21","city_16","city_114","city_160"))
dat$enrollee_id = as.factor(dat$enrollee_id)
### creating a data frame for easier use :::
frame<- tbl_df(dat)
### dealing with missing values to reduce the imbalance in the data::
frame$gender[frame$gender == ""] <- NA
frame$enrolled_university[frame$enrolled_university == ""] <- NA
frame$education_level[frame$education_level == ""] <- NA
frame$major_discipline[frame$major_discipline == ""] <- NA
frame$company_size[frame$company_size == ""] <- NA
frame$company_type[frame$company_type == ""] <- NA
frame$last_new_job[frame$last_new_job == ""] <- NA
```

Creating a SVM using the target variable to predict whether one is a potential candidate or not. Using the gamma I got from the training data: Using the linear model first::

```r
#### creating a SVM function using target to see whether one is a potential
#### candidate or not.

svmf = svm(target~., data = dat, kernel="radial",cost=1, gamma= 5.177323e-05)
```

```
> svmf = svm(target~., data = dat, kernel="radial",cost=1, gamma= 5.177323e-05)
> summary(svmf)

Call:
svm(formula = target ~ ., data = dat, kernel = "radial", cost = 1, gamma = 5.177323e-05)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  5.177323e-05
    epsilon:  0.1


Number of Support Vectors:  6115
```

To tune the SVM model, I tried to use the following range. However, This tune process took a long time on my laptop, so I use the cost to be 0.001, 1 and 10 in order to see the results better.

```
> tune.out <- tune(svm , target~., data = frame , kernel = "linear",ranges = list(cost = c(0.001 , 0
  .01, 0.1, 1, 5, 10, 100)))
```

When using cost = 1, I got the following (linear kernel) :

```
> tune.out <- tune(svm , target~., data = frame , kernel = "linear",cost = 1)
> summary(tune.out)

Error estimation of 'svm' using 10-fold cross validation: 0.1160772

> View(tune.out)
> tune.out$best.model

Call:
best.tune(method = svm, train.x = target ~ ., data = frame, kernel = "linear",
    cost = 1)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  1
      gamma:  5.177323e-05
    epsilon:  0.1


Number of Support Vectors:  5206
```

The error estimation was 0.1160772 and support vectors were 5206.
When using cost = 0.01, I got the following (linear kernel):

```
> tune.out <- tune(svm , target~., data = frame , kernel = "linear",cost = 0.001)
> summary(tune.out)

Error estimation of 'svm' using 10-fold cross validation: 0.1447447

> summary(tune.out$best.model)

Call:
best.tune(method = svm, train.x = target ~ ., data = frame, kernel = "linear",
    cost = 0.001)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  0.001
      gamma:  5.177323e-05
    epsilon:  0.1


Number of Support Vectors:  3326
```

The error estimation was 0.1447447 and support vectors were 3326.

When using cost = 10, I got the following (linear kernel) :

```
> tune.out <- tune(svm , target~., data = frame , kernel = "linear",cost = 10)
> summary(tune.out$best.model)

Call:
best.tune(method = svm, train.x = target ~ ., data = frame, kernel = "linear",
    cost = 10)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  linear
       cost:  10
      gamma:  5.177323e-05
    epsilon:  0.1


Number of Support Vectors:  5393




> summary(tune.out)

Error estimation of 'svm' using 10-fold cross validation: 0.1158676
```

The error estimation was 0.1158676 and support vectors were 5393.
Upon using the models and testing with radials as well (radial kernel):

```
> tune.out<- tune(svm,target~., data=dat,kernel = "radial",ranges = list(cost= c(0.1,1,5),
 gamma= c(0.1,5)))

> tune.out <- tune(svm , target~., data = frame , kernel = "radial",cost = 1, gamma=5)
```

```
> tune.out <- tune(svm , target~., data = frame , kernel = "radial",cost = 1, gamma=5)
> summary(tune.out)

Error estimation of 'svm' using 10-fold cross validation: 0.1585213

> summary(tune.out$best.model)

Call:
best.tune(method = svm, train.x = target ~ ., data = frame, kernel = "radial",
    cost = 1, gamma = 5)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  5
    epsilon:  0.1


Number of Support Vectors:  5426
```

Conclusion from the radial and linear model :
I was able to find that the tuning parameter using a linear model and cost as 0.001 was fastest and was able to give The error estimation as 0.1447447 which is a 14% misclassification error in the training data.

After selecting the method, I was able to get the best model For receiving the error rate and the table of predictions.

```
bestmod<- tune.out$best.model
train <- sample(1:nrow(frame),nrow(frame)/2)
xtest <- frame[-train,]
ytest<- frame$target
bestmod<- tune.out$best.model
ypred <- predict(bestmod,xtest)
ytest<- ytest[c(1:length(ypred)]
testdat<- data.frame(x = xtest,y= as.factor(ytest))
```

Thus, the best choice of the parameters will be using the linear model with a cost of 0.001 with an error rate of 14% being given.