—

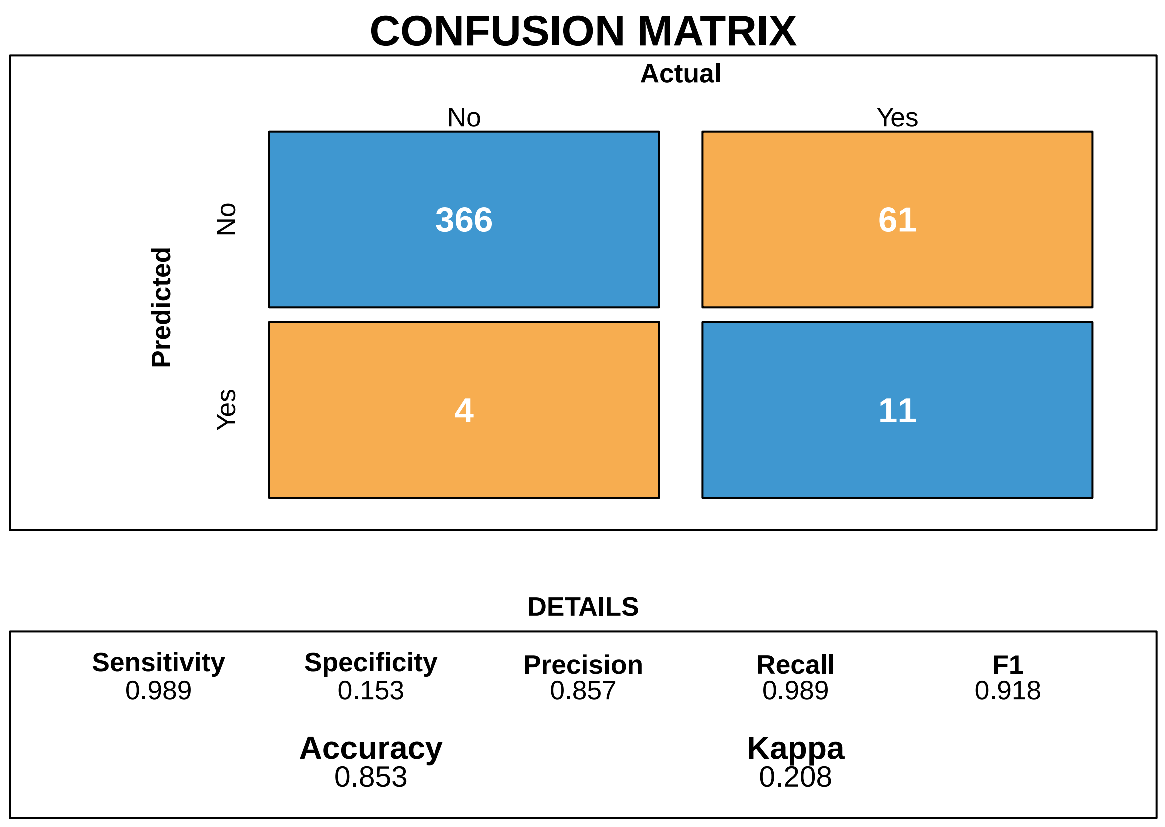
Data preparation

The code for this part is left out from the PDF due to its length…

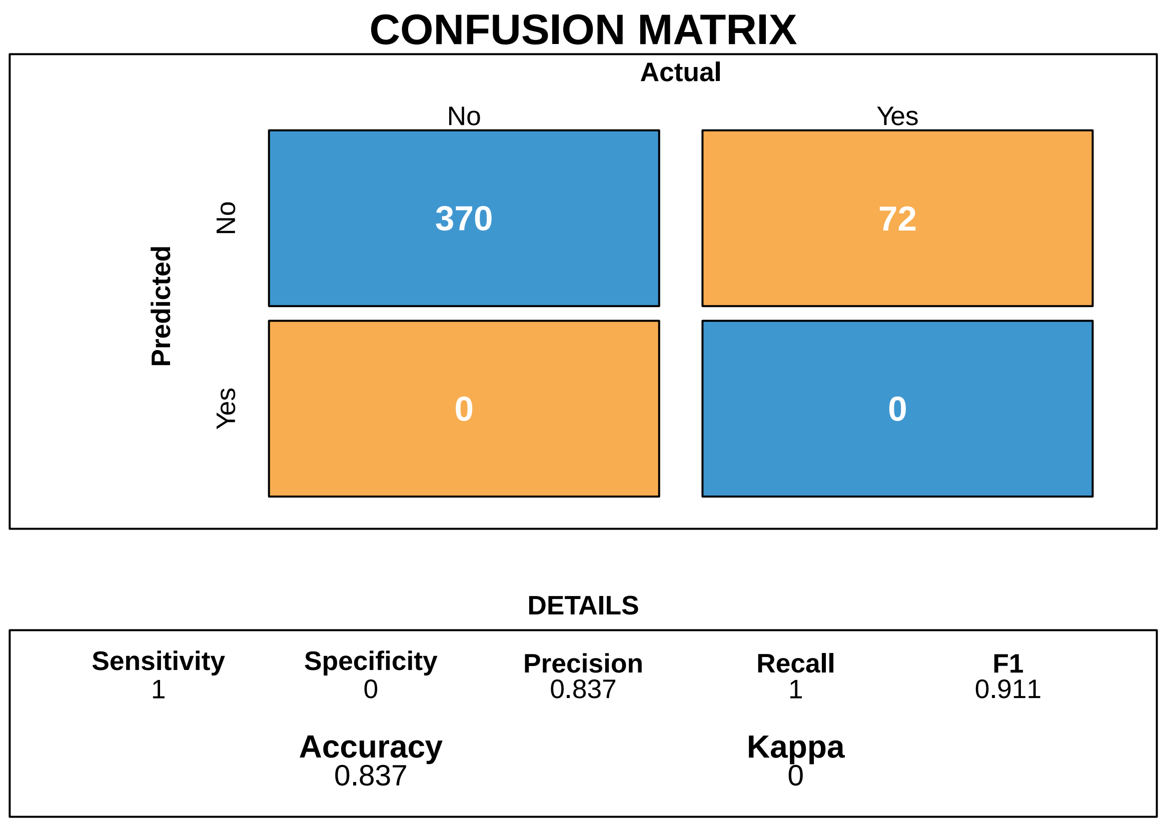
Data Cleaning

SVM Cross Validation

```
set.seed(123)
cols <- c("Attrition","EnvironmentSatisfaction","NumCompaniesWorked","JobSatisfaction","BusinessTravel"

emp_data <- emp_attrition[cols]

#Split
split <- initial_split(emp_data, prop = 0.7, strata = 'Attrition')
#Create training and test set
training <- training(split)
testing <- testing(split)

trctrl <- trainControl(method = "repeatedcv", number=10, repeats = 3)
svm_radial <- train(Attrition ~ ., data = training, method = "svmRadial", trControl=trctrl, preProcess =
svm_linear <- train(Attrition ~ ., data = training, method = "svmLinear", trControl=trctrl, preProcess =

test_pred_radial <- predict(svm_radial, newdata = testing)
test_pred_linear <- predict(svm_linear, newdata = testing)

cm_radial <- confusionMatrix(table(test_pred_radial, testing$Attrition))
cm_linear <- confusionMatrix(table(test_pred_linear, testing$Attrition))

draw_confusion_matrix(cm_radial)
```



```
draw_confusion_matrix(cm_linear)
```



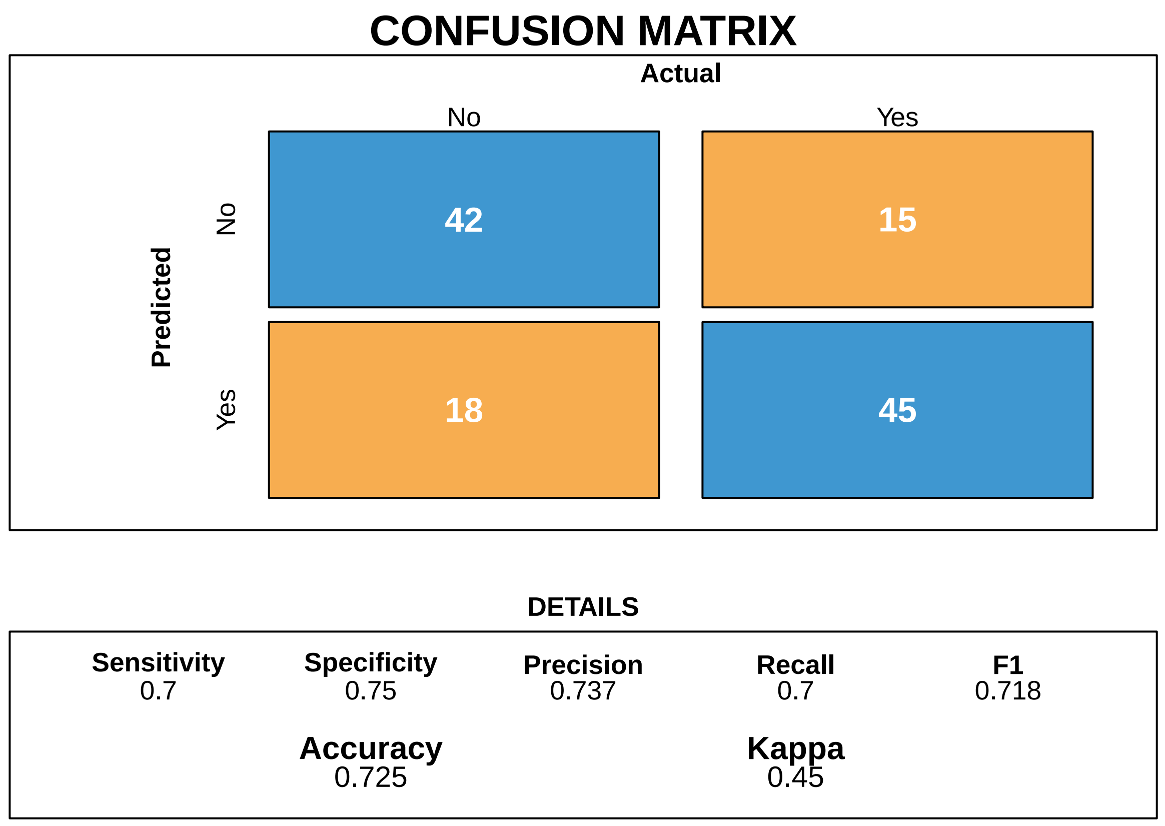
Above two model shows that the model is quite biased towards predicting “No” Attrition. This can be assumed due to the unbalanced dataset where about 80% of the dataset included employees that did not have Attrition and 20% who had Attrition.

Even though the accuracy of two model is relatively high scoring 85% and 83% this model is not approperate to predict employees that will have Attrition due to low specificity.

```
emp_data %>% count(Attrition)
```

```
##   Attrition    n
## 1        No 1233
## 2       Yes  237
```

```
new_emp<- emp_data %>% group_by(Attrition) %>% slice_sample(n=200)

#Split
split2 <- initial_split(new_emp, prop = 0.7, strata = 'Attrition')
#Create training and test set
training2 <- training(split2)
testing2 <- testing(split2)

trctrl <- trainControl(method = "repeatedcv", number=10, repeats = 3)
svm_radial2 <- train(Attrition ~ ., data = training2, method = "svmRadial", trControl=trctrl, preProcess
svm_linear2 <- train(Attrition ~ ., data = training2, method = "svmLinear", trControl=trctrl, preProcess

test_pred_radial2 <- predict(svm_radial2, newdata = testing2)
test_pred_linear2 <- predict(svm_linear2, newdata = testing2)

cm_radial2 <- confusionMatrix(table(test_pred_radial2, testing2$Attrition))
cm_linear2 <- confusionMatrix(table(test_pred_linear2, testing2$Attrition))

draw_confusion_matrix(cm_radial2)
```



CONFUSION MATRIX

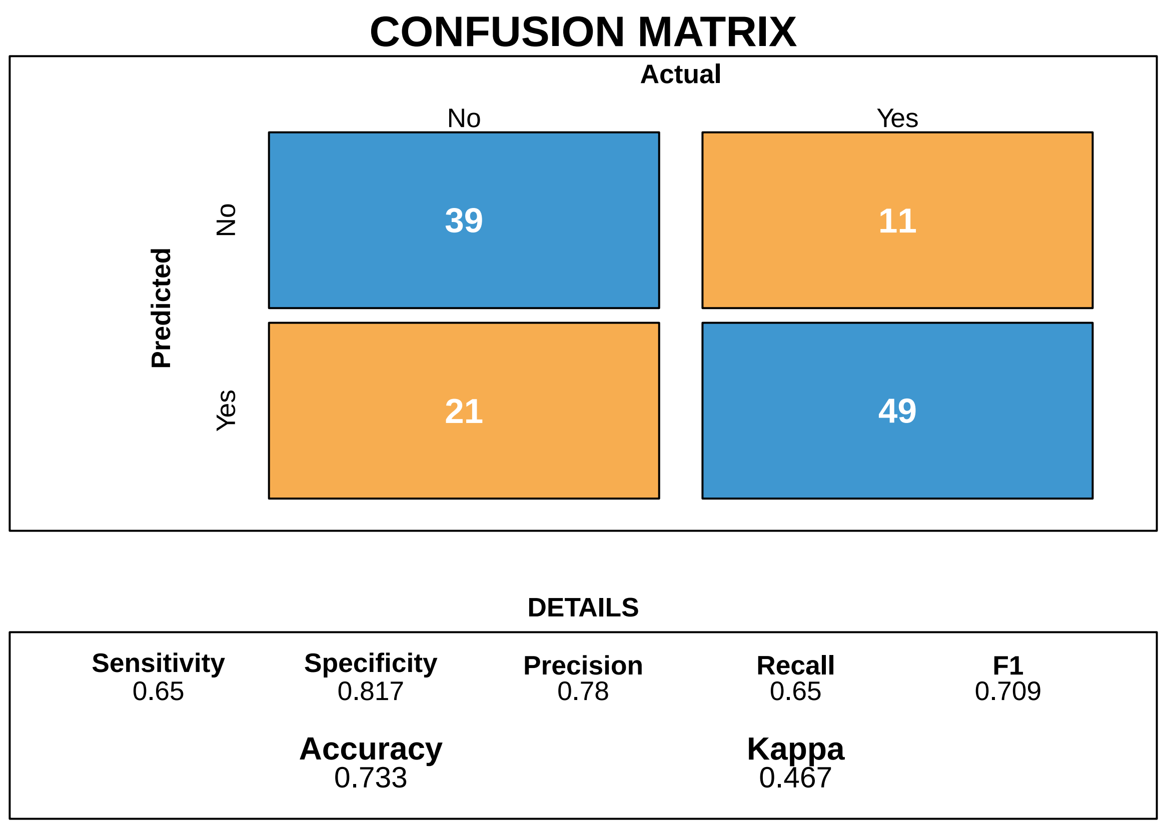
Predicted \ Actual	No	Yes
No	42	15
Yes	18	45

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.7	0.75	0.737	0.7	0.718

Accuracy	Kappa
0.725	0.45

```
draw_confusion_matrix(cm_linear2)
```



CONFUSION MATRIX

Predicted \ Actual	No	Yes
No	39	11
Yes	21	49

DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.65	0.817	0.78	0.65	0.709

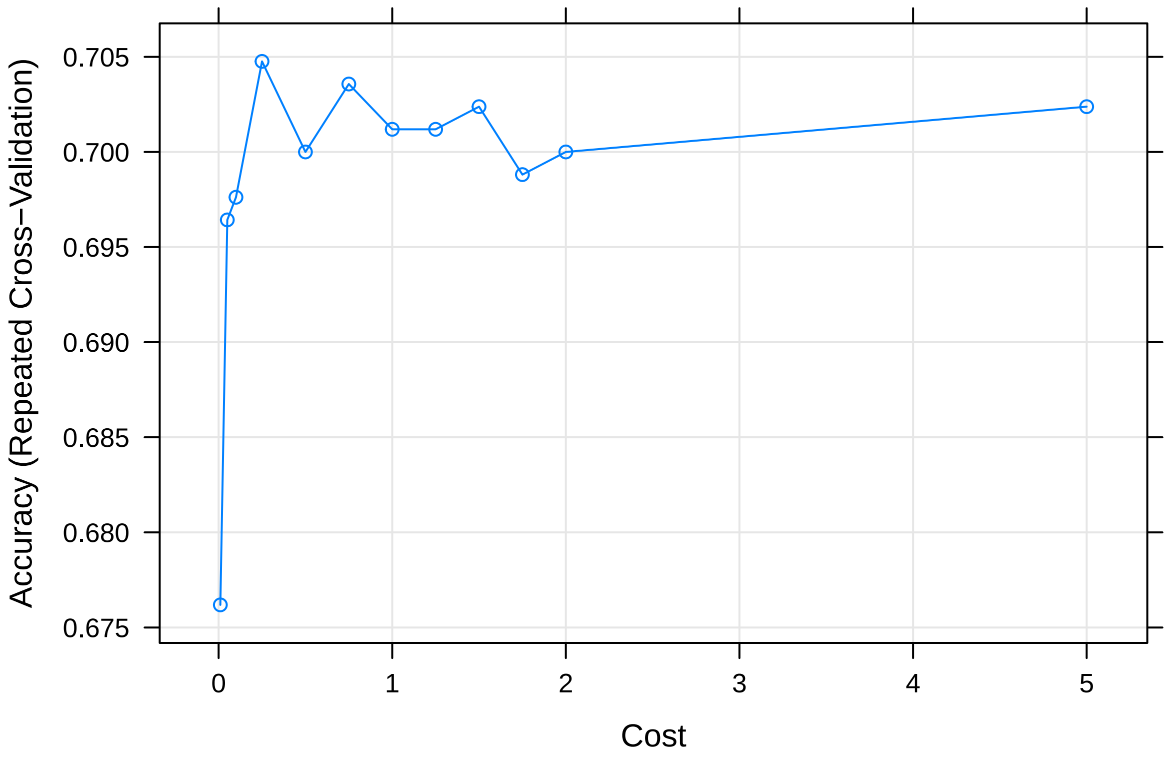
Accuracy	Kappa
0.733	0.467

After undersampling the dataset to have 50:50 ratio, the two model has lower accuracy but improved in specificity.

Model Tuning

Since we observed that dataset with 50:50 ratio showed desirable prediction output for our case, we will optimize the model by tuning the parameters.

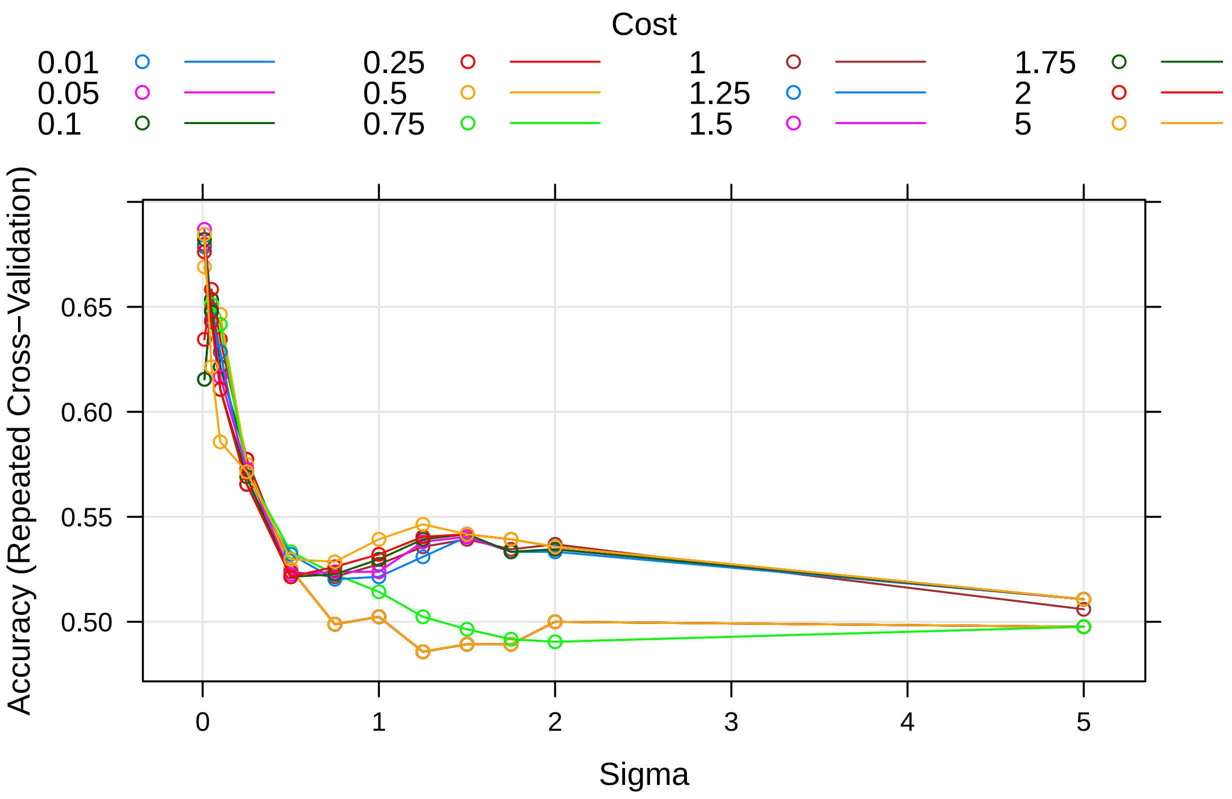
```
set.seed(123)
grid <- expand.grid(C = c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 5))

svm_linear_grid <- train(
  Attrition ~ ., data= training2, method = "svmLinear",
  trControl = trctrl,
  preProcess = c("center", "scale"),
  tuneGrid = grid,
  tuneLength = 10
)

plot(svm_linear_grid)
```



```
svm_linear_grid$bestTune
```

```
##      C
## 4 0.25
```

```
grid2 <- expand.grid(C = c(0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2, 5), sigma=c(0.01, 0

svm_radial_grid <- train(
  Attrition ~ ., data= training2, method = "svmRadial",
  trControl = trctrl,
  preProcess = c("center", "scale"),
  tuneGrid = grid2,
  tuneLength = 10
)

plot(svm_radial_grid)
```



```
svm_radial_grid$bestTune
```

```
##    sigma   C
## 97  0.01 1.5
```

The result shows C of 0.25 is the best parameter for linear SVM, for Radial SVM sigma of 0.01 and C of 1.5. Even though these values were obtained through 10 fold cross validation, the optimal parameter may change when the random seed is set to other value most likely because our dataset is not large enough and prone to random initalization of training and testing splitting.

Neural Network

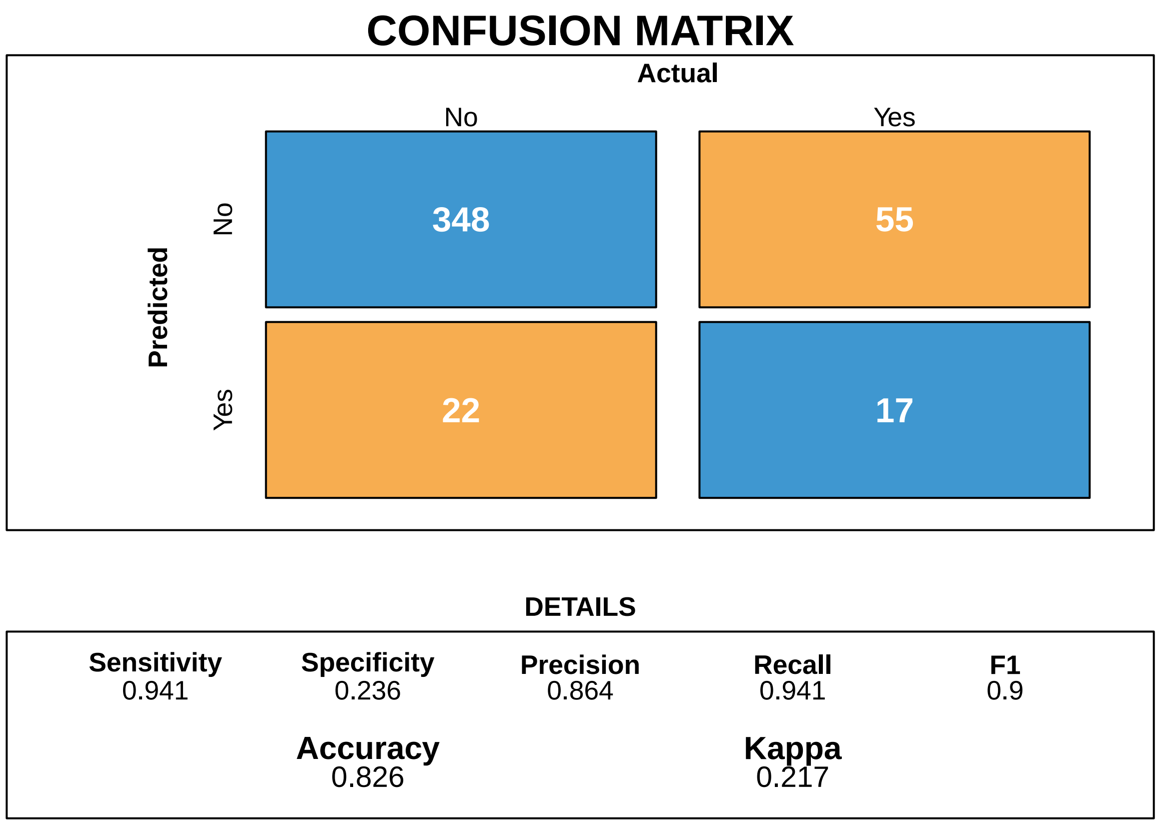
```
library(neuralnet)
```

```
##
## Attaching package: 'neuralnet'

## The following object is masked from 'package:ROCR':
##
##     prediction

## The following object is masked from 'package:dplyr':
##
##     compute
```

```
set.seed(123)
emp_net = nnet(Attrition ~ ., data=training, size=15, maxit=100, range=0.1, decay=5e-4)
```

```
## # weights:  301
## initial  value 578.352568
## iter  10 value 422.029830
## iter  20 value 381.411616
## iter  30 value 344.878275
## iter  40 value 316.048645
## iter  50 value 290.011759
## iter  60 value 280.742910
## iter  70 value 277.412237
## iter  80 value 274.046773
## iter  90 value 269.007408
## iter 100 value 260.534699
## final  value 260.534699
## stopped after 100 iterations
```

```
pred_net <- predict(emp_net, testing, type="class")
cm_nn <- confusionMatrix(table(pred=pred_net, true=testing$Attrition))
draw_confusion_matrix(cm_nn)
```



CONFUSION MATRIX

Predicted \ Actual	No	Yes
No	348	55
Yes	22	17

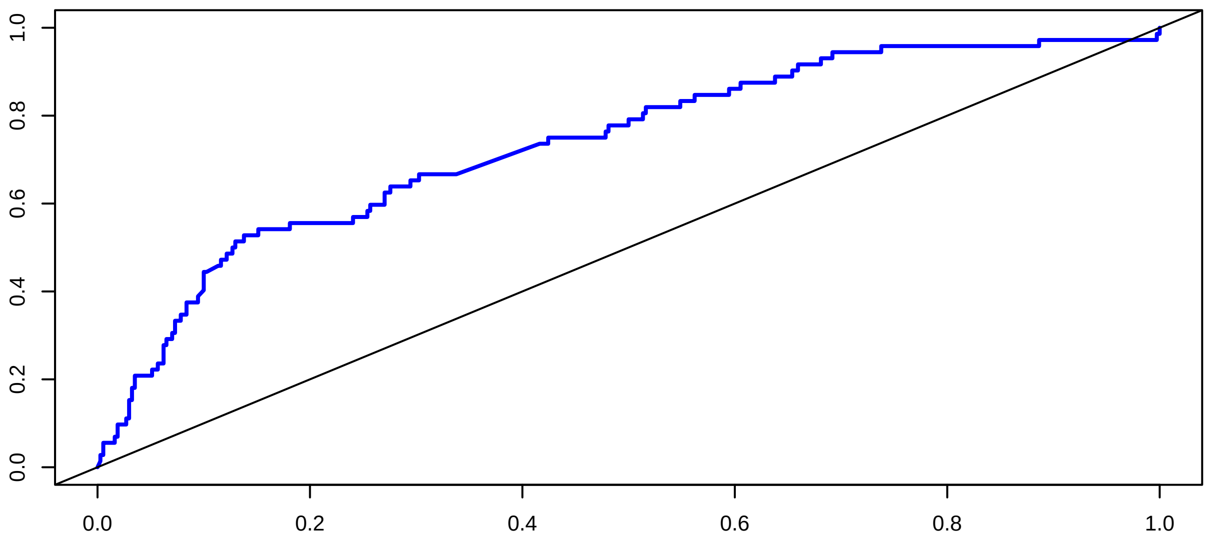
DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.941	0.236	0.864	0.941	0.9

Accuracy	Kappa
0.826	0.217

```r
pred_raw <- predict(emp_net, testing, decision.values=TRUE, type = "raw")


pred <- ROCR::prediction(pred_raw, testing$Attrition)
perf <- ROCR::performance(pred,measure="tpr", x.measure="fpr")
plot(perf, lwd=2, col="blue")
abline(a=0, b=1)
auc_ROCR <- performance(pred, measure="auc")
auc_ROCR <- auc_ROCR@y.values[[1]]
auc_ROCR
```

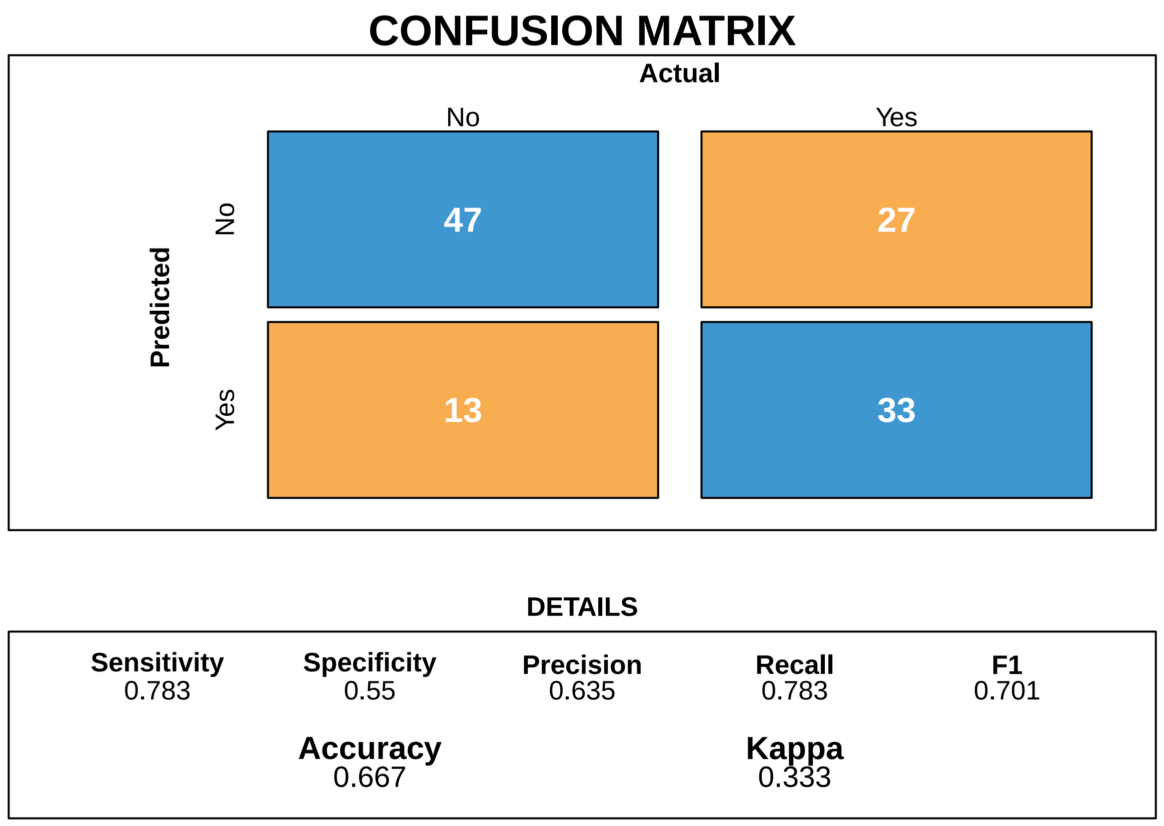
```
## [1] 0.7371246
```



```
emp_net2 = nnet(Attrition ~ ., data=training2, size=15, maxit=100, range=0.1, decay=5e-4)
```

```
## # weights:  301
## initial  value 207.110397
## iter  10 value 181.489799
## iter  20 value 171.918737
## iter  30 value 155.710904
## iter  40 value 144.243328
## iter  50 value 134.161647
## iter  60 value 129.096882
## iter  70 value 128.286932
## iter  80 value 127.540918
## iter  90 value 127.163612
## iter 100 value 126.672924
## final  value 126.672924
## stopped after 100 iterations
```

```
pred_net2 <- predict(emp_net2, testing2, type="class")
cm_nn2 <- confusionMatrix(table(pred=pred_net2, true=testing2$Attrition))
draw_confusion_matrix(cm_nn2)
```



CONFUSION MATRIX

Predicted \ Actual	No	Yes
No	47	27
Yes	13	33

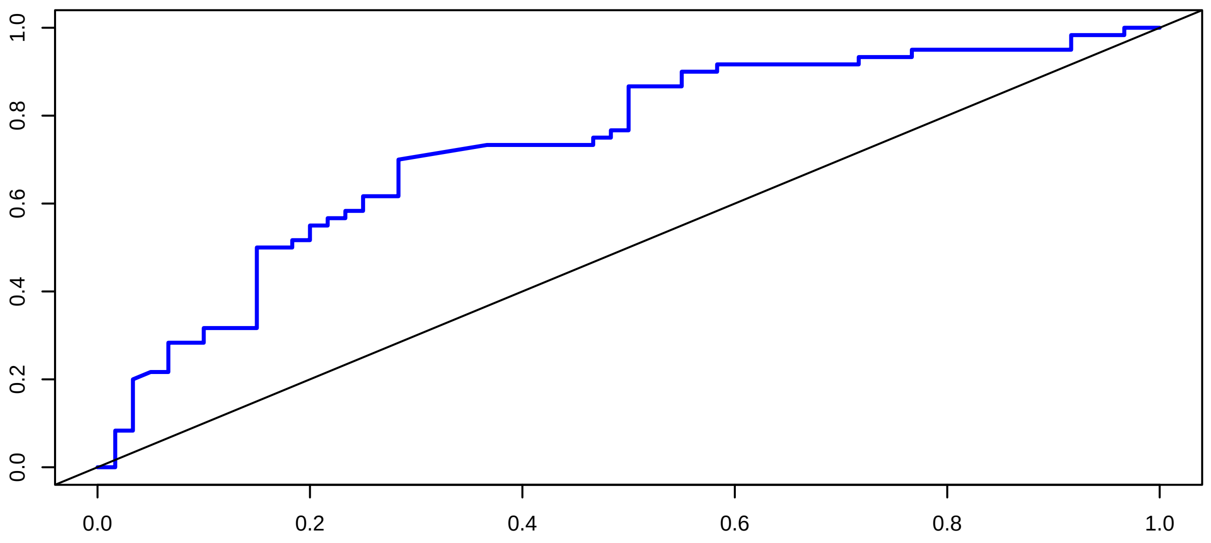
DETAILS

Sensitivity	Specificity	Precision	Recall	F1
0.783	0.55	0.635	0.783	0.701

Accuracy	Kappa
0.667	0.333

```r
pred_raw2 <- predict(emp_net2, testing2, decision.values=TRUE, type = "raw")

pred2 <- ROCR::prediction(pred_raw2, testing2$Attrition)
perf2 <- ROCR::performance(pred2, measure="tpr", x.measure="fpr")
plot(perf2, lwd=2, col="blue")
abline(a=0, b=1)
auc_ROCR2 <- performance(pred2, measure="auc")
auc_ROCR2 <- auc_ROCR2@y.values[[1]]
auc_ROCR2
```

```
## [1] 0.7334722
```



For neural network model result is interesting compare to SVM model. Even though the model is biased twoards classifying employee as not having attrition, this result can be adjusted by moving the cutoff value. Both the full sample model and under sampled model shows similar AUC value thus choosing an appropriate cutoff value will allow the model to predict attrition of employee with desirable sensitivity and specificity.