Diebold Model building

Libraries required

```
knitr::opts_chunk$set(eval = TRUE)
```

```
library(keras)
library(dplyr)
library(magrittr)
library(caret)
library(pROC)
citation("pROC")
```

```
##
## If you use pROC in published research, please cite the following
## paper:
##
##   Xavier Robin, Natacha Turck, Alexandre Hainard, Natalia Tiberti,
##   Frédérique Lisacek, Jean-Charles Sanchez and Markus Müller
##   (2011). pROC: an open-source package for R and S+ to analyze and
##   compare ROC curves. BMC Bioinformatics, 12, p. 77.  DOI:
##   10.1186/1471-2105-12-77
##   <http://www.biomedcentral.com/1471-2105/12/77/>
##
## A BibTeX entry for LaTeX users is
##
##   @Article{,
##     title = {pROC: an open-source package for R and S+ to analyze and compare ROC cur
ves},
##     author = {Xavier Robin and Natacha Turck and Alexandre Hainard and Natalia Tibert
i and Frédérique Lisacek and Jean-Charles Sanchez and Markus Müller},
##     year = {2011},
##     journal = {BMC Bioinformatics},
##     volume = {12},
##     pages = {77},
##   }
##
## The authors would be glad to hear how pROC is employed. You are
## kindly encouraged to notify Xavier Robin
## <pROC-cran@xavier.robin.name> about any work you publish.
```

Splitting the data into train, validation

```
set.seed(123)
sample_train<- sample(seq_len(nrow(invoiced)), size = floor(0.60*nrow(invoiced)))
sample_valid<- sample(seq_len(nrow(invoiced)), size = floor(0.20*nrow(invoiced)))
sample_test <- sample(seq_len(nrow(invoiced)), size = floor(0.20*nrow(invoiced)))
```

```
train_call_text <- free_form_call_text[sample_train, ] %>% as.array()
val_call_text <- free_form_call_text[sample_valid, ] %>% as.array()
test_call_text <- free_form_call_text[sample_test, ] %>% as.array()
 dim(train_call_text)
```

```
## [1] 84767   200
```

```
 dim(val_call_text)
```

```
## [1] 28255   200
```

```
 dim(test_call_text)
```

```
## [1] 28255   200
```

```
train_billing_notes <- free_form_billing_notes[sample_train, ] %>% as.array()
val_billing_notes <- free_form_billing_notes[sample_valid, ] %>% as.array()
test_billing_notes <- free_form_billing_notes[sample_test, ] %>% as.array()
dim(train_billing_notes)
```

```
## [1] 84767   200
```

```
dim(val_billing_notes)
```

```
## [1] 28255   200
```

```
dim(test_billing_notes)
```

```
## [1] 28255   200
```

```
train_categorical_data <- categorical_data[sample_train, ] %>% as.array()
val_categorical_data <-  categorical_data[sample_valid, ] %>% as.array()
test_categorical_data <- categorical_data[sample_test, ] %>% as.array()
dim(train_categorical_data)
```

```
## [1] 84767   337
```

```
dim(val_categorical_data)
```

```
## [1] 28255   337
```

```
dim(test_categorical_data)
```

```
## [1] 28255   337
```

```
train_invoiced <- invoiced[sample_train, ] %>% as.array()
val_invoiced <- invoiced[sample_valid, ] %>% as.array()
test_invoiced <- invoiced[sample_test, ] %>% as.array()
```

Merging Multiple Inputs

```
call_text_layer <- layer_input(shape = c(CONSTANTS$MAX_LEN), name = "call_text_layer")
billing_notes_layer <- layer_input(shape = c(CONSTANTS$MAX_LEN),name = "billing_notes_la
yer")
categorical_layer_model <- layer_input(shape = c(dim(categorical_data)[2]), name = 'cate
gorica_layer_model')
```

Creating The Embedding layers

```
call_text_embedding <- call_text_layer %>% layer_embedding(input_dim = CONSTANTS$MAX_WOR
DS, output_dim = 512, input_length = CONSTANTS$MAX_LEN, name = "call_text_embedding") %
>% layer_dropout(0.6) %>% layer_flatten()
billing_notes_embedding <- billing_notes_layer %>% layer_embedding(input_dim = CONSTANTS
$MAX_WORDS,output_dim = 512,input_length = CONSTANTS$MAX_LEN, name = "billing_notes_embe
dding") %>% layer_dropout(0.6) %>% layer_flatten()
```

Input and Auxilary Layers

```
main_output <- layer_concatenate(c(call_text_embedding, billing_notes_embedding, categor
ical_layer_model)) %>%
  layer_dense(units = 64, activation = 'relu', kernel_regularizer =  regularizer_l2(0.00
1)) %>%
  layer_dense(units = 64, activation = 'relu', kernel_regularizer =  regularizer_l2(0.00
1)) %>%
  layer_dense(units = 64, activation = 'relu', kernel_regularizer =  regularizer_l2(0.00
1)) %>%
  layer_dense(units = 1, activation = 'sigmoid', name = 'main_output')
```

Model Building

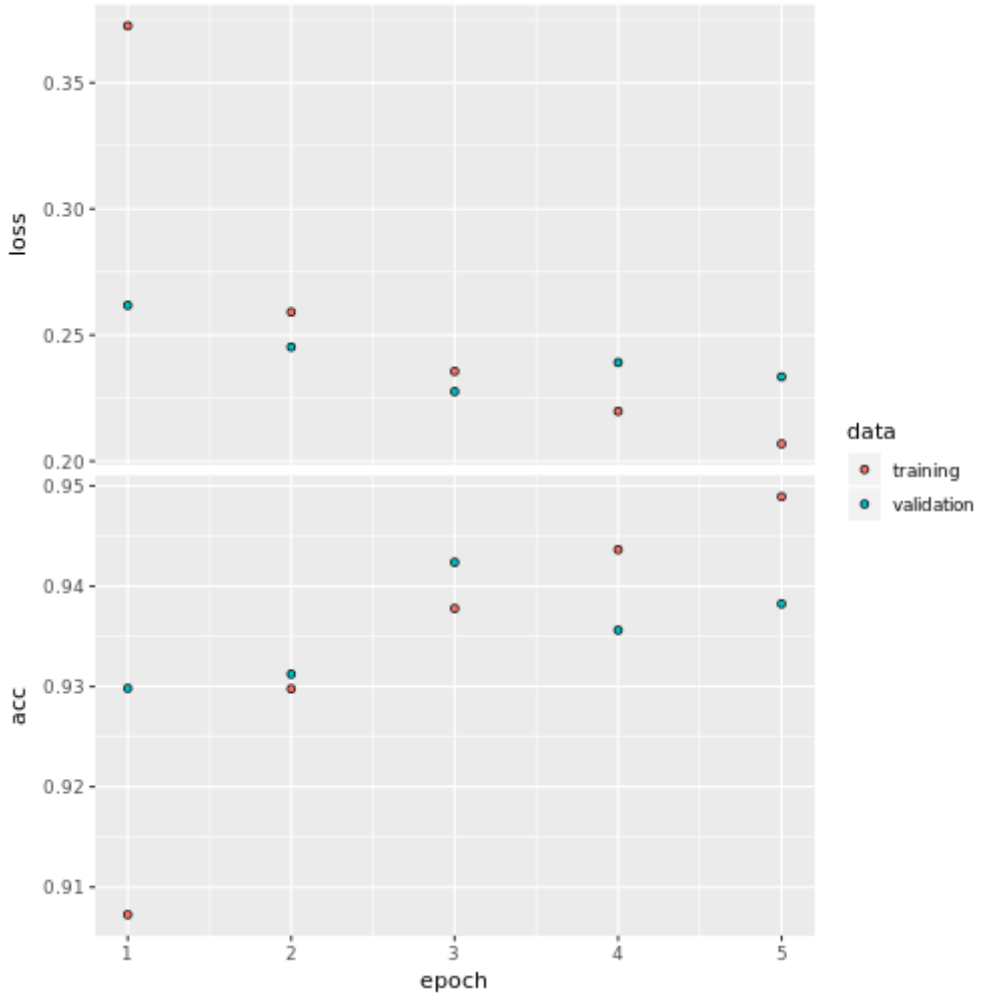
```
model <- keras_model(inputs = c(call_text_layer, billing_notes_layer, categorical_layer_
model), outputs = main_output)
```

```
model %>% compile(optimizer = 'rmsprop', loss = 'binary_crossentropy',metric = 'accurac
y')
summary(model)
```

```
## ____________________________________________________________________________
## Layer (type)            Output Shape     Param #  Connected to
## ============================================================================
## call_text_layer (InputL (None, 200)      0
## ____________________________________________________________________________
## billing_notes_layer (In (None, 200)      0
## ____________________________________________________________________________
## call_text_embedding (Em (None, 200, 512) 10240000 call_text_layer[0][0]
## ____________________________________________________________________________
## billing_notes_embedding (None, 200, 512) 10240000 billing_notes_layer[0][0]
## ____________________________________________________________________________
## dropout (Dropout)       (None, 200, 512) 0        call_text_embedding[0][0]
## ____________________________________________________________________________
## dropout_1 (Dropout)     (None, 200, 512) 0        billing_notes_embedding[0
## ____________________________________________________________________________
## flatten (Flatten)       (None, 102400)   0        dropout[0][0]
## ____________________________________________________________________________
## flatten_1 (Flatten)     (None, 102400)   0        dropout_1[0][0]
## ____________________________________________________________________________
## categorica_layer_model  (None, 337)      0
## ____________________________________________________________________________
## concatenate (Concatenat (None, 205137)   0        flatten[0][0]
##                                                  flatten_1[0][0]
##                                                  categorica_layer_model[0]
## ____________________________________________________________________________
## dense (Dense)           (None, 64)       13128832 concatenate[0][0]
## ____________________________________________________________________________
## dense_1 (Dense)         (None, 64)       4160     dense[0][0]
## ____________________________________________________________________________
## dense_2 (Dense)         (None, 64)       4160     dense_1[0][0]
## ____________________________________________________________________________
## main_output (Dense)     (None, 1)        65       dense_2[0][0]
## ============================================================================
## Total params: 33,617,217
## Trainable params: 33,617,217
## Non-trainable params: 0
## ____________________________________________________________________________
```

```
history_model <- model %>% fit(x = list(train_call_text, train_billing_notes, train_cate
gorical_data), y = train_invoiced, epochs = 5, batch_size = 128, validation_data = list
(list(val_call_text, val_billing_notes, val_categorical_data),val_invoiced))
```

```
plot(history_model)
```



plot of chunk unnamed-chunk-18

Testing the accuracy of the model on the test set

```
result <- model %>% evaluate(list(test_call_text, test_billing_notes, test_categorical_d
ata), test_invoiced)
result$loss
result$acc
```

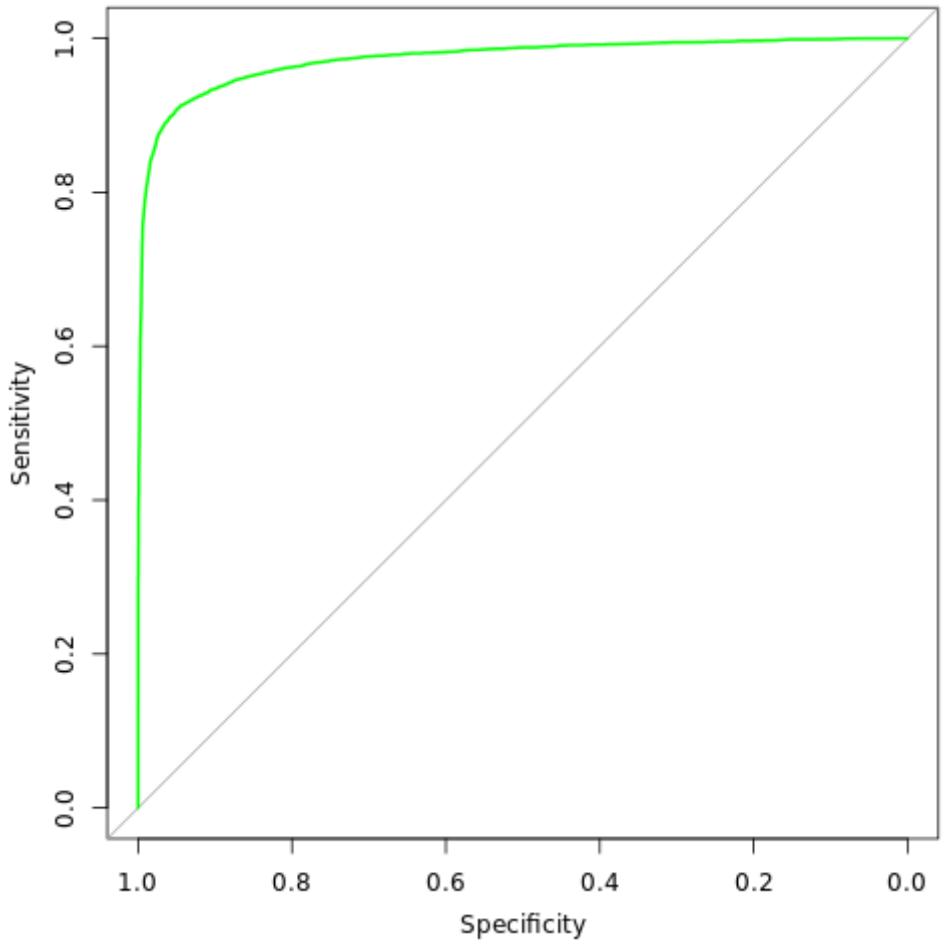
Model ROC

```
probability <- predict(model, list(test_call_text, test_billing_notes, test_categorical_
data), batch_size = 128)
roc <- roc(test_invoiced, as.vector(probability))
roc
```

```
##
## Call:
## roc.default(response = test_invoiced, predictor = as.vector(probability))
##
## Data: as.vector(probability) in 22965 controls (test_invoiced 0) < 5290 cases (test_i
nvoiced 1).
## Area under the curve: 0.9739
```

Model AUC

```
plot(roc, col='green')
```



plot of chunk unnamed-chunk-21

Confusion matrix

```
class_prediction <- as.numeric(probability > .30) %>% as.factor()
confusionMatrix(class_prediction, as.factor(test_invoiced), mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 20648   346
##          1  2317  4944
##
##                Accuracy : 0.9058
##                  95% CI : (0.9023, 0.9091)
##     No Information Rate : 0.8128
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7292
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##               Precision : 0.9835
##                  Recall : 0.8991
##                      F1 : 0.9394
##              Prevalence : 0.8128
##          Detection Rate : 0.7308
##    Detection Prevalence : 0.7430
##       Balanced Accuracy : 0.9169
##
##        'Positive' Class : 0
##
```