Urdu Language Sentiment Classifier

Dan S. Reznik

July, 2020

# Description of Problem

* *Background*: A large multinational corporation is seeking to automatically identify the sentiment that their customer base talks about on social media. They would like to expand this capability into multiple languages. Many 3rd party tools exist for sentiment analysis, however, they need help with under-resourced languages.
* *Goal*: Train a sentiment classifier (Positive, Negative, Neutral) on a corpus of the provided documents. Your goal is to maximize accuracy. There is special interest in being able to accurately detect negative sentiment. The training data includes documents from a wide variety of sources, not merely social media, and some of it may be inconsistently labeled. Please describe the business outcomes in your work sample including how data limitations impact your results and how these limitations could be addressed in a larger project.
* *Data*: Link to data: <http://archive.ics.uci.edu/ml/datasets/Roman+Urdu+Data+Set>

## Outline of the solution:

1. Analyze data, data quality, cleanse data
2. Will go for a “bag of words” (orderless) approach. Create document term matrix (rows are the documents, columns are non-sparse terms)
3. Filter out “neutrals”, only keep “negative” and “positive” documents
4. Train multiple classifier models (H2O automl) with double weights on the negative training examples, use “misclassification” as the functional to optimize. Positive+Neutral are lumped
5. Produce a leaderboard, report metrics, e.g., our 5-CV AUC was ~75%, and confusion matrix shows error rate for negatives of only 13%.
6. Suggestions for improvements to the appearin the end

# Analyze input file

Load packages

library(tidyverse)  
library(data.table)  
library(tm)  
library(xgboost)  
library(h2o)

Source file

fname <- "data/Roman Urdu DataSet.csv"

Encoding is UTF-8

guess\_encoding(fname)

## # A tibble: 3 x 2  
## encoding confidence  
## <chr> <dbl>  
## 1 UTF-8 1   
## 2 windows-1252 0.290  
## 3 windows-1254 0.290

Look at the first few lines of file:

* comma separated, header is missing

read\_lines(fname,n\_max = 3)

## [1] "Sai kha ya her kisi kay bus ki bat nhi hai lakin main ki hal kal bi Aj aur aj bi sirf Aus say bus,Positive,"  
## [2] "sahi bt h,Positive,"   
## [3] "\"Kya bt hai,\",Positive,"

Reads 3-col csv as characters

df\_urdu <- read\_csv("data/Roman Urdu DataSet.csv",col\_names = c("phrase","sentiment","bogus"),  
 col\_types = "ccc", # all are chars  
 #n\_max=3  
 )  
df\_urdu %>% glimpse

## Rows: 20,229  
## Columns: 3  
## $ phrase <chr> "Sai kha ya her kisi kay bus ki bat nhi hai lakin main ki...  
## $ sentiment <chr> "Positive", "Positive", "Positive", "Positive", "Positive...  
## $ bogus <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...

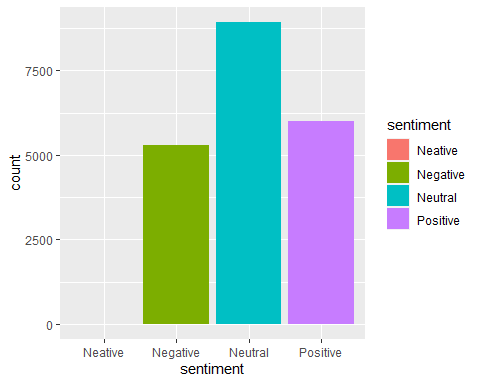
Third column can truly be ignored

df\_urdu %>% count(bogus,sort=T)

## # A tibble: 7 x 2  
## bogus n  
## <chr> <int>  
## 1 <NA> 20222  
## 2 till here 2  
## 3 ------ 1  
## 4 ------- 1  
## 5 ---------- 1  
## 6 ---------------- 1  
## 7 9090 1

Categories on sentiment column:

df\_urdu %>%  
 ggplot(aes(sentiment,fill=sentiment)) +  
 geom\_bar()



One sacred line with “Neative”, what is it?

df\_urdu %>% filter(sentiment == "Neative")

## # A tibble: 1 x 3  
## phrase sentiment bogus  
## <chr> <chr> <chr>  
## 1 product achi hai but wrong waist size send kar diya. Neative <NA>

A few lines have NA phrases, which need to be removed.

df\_urdu %>%  
 filter(is.na(phrase))

## # A tibble: 113 x 3  
## phrase sentiment bogus  
## <chr> <chr> <chr>  
## 1 <NA> Neutral <NA>   
## 2 <NA> Neutral <NA>   
## 3 <NA> Neutral <NA>   
## 4 <NA> Neutral <NA>   
## 5 <NA> Neutral <NA>   
## 6 <NA> Neutral <NA>   
## 7 <NA> Neutral <NA>   
## 8 <NA> Neutral <NA>   
## 9 <NA> Neutral <NA>   
## 10 <NA> Neutral <NA>   
## # ... with 103 more rows

## Study characters in the set

df\_urdu$phrase[1] %>% str\_split("")

## [[1]]  
## [1] "S" "a" "i" " " "k" "h" "a" " " "y" "a" " " "h" "e" "r" " " "k" "i" "s" "i"  
## [20] " " "k" "a" "y" " " "b" "u" "s" " " "k" "i" " " "b" "a" "t" " " "n" "h" "i"  
## [39] " " "h" "a" "i" " " "l" "a" "k" "i" "n" " " "m" "a" "i" "n" " " "k" "i" " "  
## [58] "h" "a" "l" " " "k" "a" "l" " " "b" "i" " " "A" "j" " " "a" "u" "r" " " "a"  
## [77] "j" " " "b" "i" " " "s" "i" "r" "f" " " "A" "u" "s" " " "s" "a" "y" " " "b"  
## [96] "u" "s"

Frequency count of all chars used

count\_chars <- function(df,col) {  
 df %>% # head(10) %>%  
 mutate(chars = {{col}} %>% str\_split("")) %>%  
 select(chars) %>%  
 unnest(chars) %>%  
 count(chars,sort=T) %>%  
 mutate(prop=sprintf("%.3f",n/sum(n)))   
}

df\_char\_freq <- df\_urdu %>% count\_chars(phrase)  
df\_char\_freq%>%glimpse

## Rows: 272  
## Columns: 3  
## $ chars <chr> " ", "a", "i", "e", "h", "n", "r", "k", "t", "o", "s", "m", "...  
## $ n <int> 251023, 193318, 89650, 82255, 79466, 63038, 57748, 52567, 470...  
## $ prop <chr> "0.183", "0.141", "0.065", "0.060", "0.058", "0.046", "0.042"...

Which are non-alpha. Note 0x001F602 doesn’t exist.

df\_char\_freq %>% filter(!str\_detect(chars,"[:alpha:]"))

## # A tibble: 183 x 3  
## chars n prop   
## <chr> <int> <chr>  
## 1 " " 251023 0.183  
## 2 "." 10159 0.007  
## 3 "," 2761 0.002  
## 4 "1" 2466 0.002  
## 5 "0" 1633 0.001  
## 6 "?" 1599 0.001  
## 7 "9" 1584 0.001  
## 8 "\U0001f602" 1506 0.001  
## 9 "2" 1357 0.001  
## 10 ":" 959 0.001  
## # ... with 173 more rows

Preprocessing steps:

* eliminate NA phrases
* change ‘Neative’ to ‘Negative’
* eliminate non-alpha (includes numbers)
* convert all to lower-case
* squish multiple spaces

df\_urdu\_clean <- df\_urdu %>%  
 filter(!is.na(phrase)) %>%  
 mutate(sentiment=if\_else(sentiment=="Neative","Negative",sentiment)) %>%  
 mutate(phrase\_clean=phrase %>%  
 str\_remove\_all("[^ [:alpha:]]") %>%  
 str\_to\_lower() %>%  
 str\_squish()) %>%  
 select(phrase\_clean,sentiment)  
df\_urdu\_clean %>% head(10)

## # A tibble: 10 x 2  
## phrase\_clean sentiment  
## <chr> <chr>   
## 1 sai kha ya her kisi kay bus ki bat nhi hai lakin main ki hal kal b~ Positive   
## 2 sahi bt h Positive   
## 3 kya bt hai Positive   
## 4 wah je wah Positive   
## 5 are wha kaya bat hai Positive   
## 6 wah kya baat likhi Positive   
## 7 wha itni sari khubiya Positive   
## 8 itni khubiya Positive   
## 9 ya allah rehm farma hm sab pe or zalimo ko hidayat de ameen Positive   
## 10 please everyone allah swt ka naam hamesha bary lawzo main likha ka~ Positive

Confirm chars are ok

df\_urdu\_clean %>% count\_chars(phrase\_clean)

## # A tibble: 62 x 3  
## chars n prop   
## <chr> <int> <chr>  
## 1 " " 241819 0.182  
## 2 "a" 201695 0.152  
## 3 "i" 92547 0.070  
## 4 "e" 83985 0.063  
## 5 "h" 83624 0.063  
## 6 "n" 65396 0.049  
## 7 "r" 59609 0.045  
## 8 "k" 58021 0.044  
## 9 "t" 49364 0.037  
## 10 "s" 48592 0.037  
## # ... with 52 more rows

# Token-oriented study

df\_urdu\_tokens <- df\_urdu\_clean %>%  
 mutate(token = str\_split(phrase\_clean,fixed(" "))) %>%  
 select(token) %>%  
 unnest(token) %>%  
 count(token,sort=T) %>%  
 mutate(id=row\_number(),  
 prop=n/sum(n),  
 propSum=cumsum(prop))  
df\_urdu\_tokens

## # A tibble: 32,734 x 5  
## token n id prop propSum  
## <chr> <int> <int> <dbl> <dbl>  
## 1 ki 5758 1 0.0220 0.0220  
## 2 ke 5354 2 0.0204 0.0424  
## 3 mein 4361 3 0.0166 0.0591  
## 4 hai 3913 4 0.0149 0.0740  
## 5 ka 3586 5 0.0137 0.0877  
## 6 ko 3572 6 0.0136 0.101   
## 7 se 3237 7 0.0124 0.114   
## 8 aur 3173 8 0.0121 0.126   
## 9 k 2810 9 0.0107 0.137   
## 10 ne 2055 10 0.00785 0.144   
## # ... with 32,724 more rows

## Build Document Term Matrix

Create term matrix. Each rows has the count of the top N tokens

corpus\_urdu <- SimpleCorpus(VectorSource(df\_urdu\_clean$phrase\_clean))

Creat a document term matrix

fn\_tf\_idf <- function(x) weightTfIdf(x, normalize = F)  
  
dtm\_urdu <- DocumentTermMatrix(corpus\_urdu  
 , control = list(weighting = fn\_tf\_idf)  
 )  
dtm\_urdu

## <<DocumentTermMatrix (documents: 20116, terms: 32360)>>  
## Non-/sparse entries: 186715/650767045  
## Sparsity : 100%  
## Maximal term length: 82  
## Weighting : term frequency - inverse document frequency (tf-idf)

Inspect first five lines and first 10 columns

mtx <- inspect(dtm\_urdu[1:10,1:100]) %>% as.matrix

## <<DocumentTermMatrix (documents: 10, terms: 100)>>  
## Non-/sparse entries: 59/941  
## Sparsity : 94%  
## Maximal term length: 9  
## Weighting : term frequency - inverse document frequency (tf-idf)  
## Sample :  
## Terms  
## Docs bat bus everyone itni khubiya lawzo swt wah  
## 1 6.514696 15.27569 0.00000 0.0000 0.00000 0.00000 0.00000 0.000000  
## 10 0.000000 0.00000 12.29606 0.0000 0.00000 14.29606 13.29606 0.000000  
## 2 0.000000 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000 0.000000  
## 3 0.000000 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000 0.000000  
## 4 0.000000 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000 17.191232  
## 5 6.514696 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000 0.000000  
## 6 0.000000 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000 8.595616  
## 7 0.000000 0.00000 0.00000 7.7262 13.29606 0.00000 0.00000 0.000000  
## 8 0.000000 0.00000 0.00000 7.7262 13.29606 0.00000 0.00000 0.000000  
## 9 0.000000 0.00000 0.00000 0.0000 0.00000 0.00000 0.00000 0.000000  
## Terms  
## Docs wha zalimo  
## 1 0.00000 0.00000  
## 10 0.00000 0.00000  
## 2 0.00000 0.00000  
## 3 0.00000 0.00000  
## 4 0.00000 0.00000  
## 5 11.71109 0.00000  
## 6 0.00000 0.00000  
## 7 11.71109 0.00000  
## 8 0.00000 0.00000  
## 9 0.00000 12.71109

mtx %>% dim

## [1] 10 10

Note: 99.8% sparsity => 673 columns, AUC ~ 75% 995 => ~200 columns, AUC falls to 70%

df\_urdu\_dtm <- removeSparseTerms(dtm\_urdu, 0.998) %>%  
 as.matrix %>% as\_tibble %>%  
 bind\_cols(df\_urdu\_clean %>% select(sentiment) %>%  
 mutate\_at(vars(sentiment),as.factor)) %>%  
 select(sentiment,everything())  
df\_urdu\_dtm %>% dim

## [1] 20116 674

# (Failed Attempt 1): Linear separability

Create train (80%) and test sets.

set.seed(0)  
permutations <- sample.int(nrow(df\_urdu\_dtm),nrow(df\_urdu\_dtm))  
train\_pct <- 0.8  
train\_max <- as.integer(length(permutations)\*train\_pct)  
df\_urdu\_dtm\_train <- df\_urdu\_dtm[permutations[1:train\_max],]  
df\_urdu\_dtm\_test <- df\_urdu\_dtm[permutations[(train\_max+1):length(permutations)],]

Get prob matrix of top 100 words, only using 80% of the dataset

df\_urdu\_word\_probs <- df\_urdu\_dtm\_train %>%  
 pivot\_longer(-sentiment) %>%  
 group\_by(sentiment,name) %>%  
 summarize(value=sum(value)) %>%  
 group\_by(sentiment) %>%  
 # slice\_max(n=1000, order\_by=value) %>%  
 ungroup() %>%  
 pivot\_wider(names\_from="sentiment") %>%  
 rowwise() %>%  
 mutate(total=sum(c\_across(Negative:Positive))) %>%  
 ungroup() %>%  
 mutate\_at(vars(Negative:Positive),~./sum(.))

## `summarise()` regrouping output by 'sentiment' (override with `.groups` argument)

df\_urdu\_word\_probs

## # A tibble: 673 x 5  
## name Negative Neutral Positive total  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 aaj 0.00209 0.00190 0.00182 955.  
## 2 aala 0.000296 0.000630 0.00124 382.  
## 3 aam 0.000961 0.00100 0.00106 503.  
## 4 aane 0.000561 0.000608 0.000514 277.  
## 5 aap 0.00349 0.00483 0.00433 2110.  
## 6 aata 0.000435 0.000496 0.000744 285.  
## 7 aaya 0.000978 0.000868 0.000948 462.  
## 8 abdul 0.00104 0.000766 0.00129 520.  
## 9 abdullah 0.000680 0.000933 0.000602 364.  
## 10 abhi 0.00135 0.00179 0.00101 677.  
## # ... with 663 more rows

Create efficient lookup table

dt\_urdu\_word\_probs <- df\_urdu\_word\_probs %>% as.data.table()  
setkey(dt\_urdu\_word\_probs,"name")

Evaluate performance (create confusion matrix)

categorize\_phrase <- function(df\_phrase) {  
 dt\_pivot <- df\_phrase %>%  
 pivot\_longer(-sentiment,names\_to="name") %>%  
 as.data.table()  
   
 df\_sums <- merge(dt\_pivot,dt\_urdu\_word\_probs,by="name") %>%  
 # inner\_join(df\_urdu\_word\_probs) %>%  
 mutate\_at(vars(Negative:Positive),~.\*log(1+value)) %>%  
 summarize\_at(vars(Negative:Positive),sum,na.rm=T)  
 pred <- with(df\_sums,  
 { sum\_max <- which.max(c(Negative,Neutral,Positive))  
 c("Negative","Neutral","Positive")[sum\_max] })  
 pred  
}  
  
df\_urdu\_dtm\_test %>% head(1) %>% categorize\_phrase()

## [1] "Positive"

Slow: predict on test set

df\_urdu\_dtm\_test\_confusion\_mtx <- df\_urdu\_dtm\_test %>%  
 # head(200) %>%  
 mutate(pred=pmap\_chr(., ~categorize\_phrase(as\_tibble\_row(list(...))))) %>%  
 select(sentiment,pred) %>%  
 count(sentiment,pred) %>%  
 pivot\_wider(names\_from = "pred",values\_from="n")  
df\_urdu\_dtm\_test\_confusion\_mtx %>% write\_csv("data/simple\_confusion\_mtx\_norm\_col\_tf\_idf.csv")

## Confusion Matrix: normalize cols

read\_csv("data/simple\_confusion\_mtx\_norm\_col.csv") %>%  
 rowwise() %>%  
 mutate(total=c\_across(Negative:Positive)%>%sum) %>%  
 mutate\_at(vars(Negative:Positive),~./total)

## Parsed with column specification:  
## cols(  
## sentiment = col\_character(),  
## Negative = col\_double(),  
## Neutral = col\_double(),  
## Positive = col\_double()  
## )

## # A tibble: 3 x 5  
## # Rowwise:   
## sentiment Negative Neutral Positive total  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 Negative 0.576 0.194 0.230 1063  
## 2 Neutral 0.434 0.358 0.208 1732  
## 3 Positive 0.234 0.218 0.548 1229

## Confusion Matrix: normalize cols with tf\*idf

read\_csv("data/simple\_confusion\_mtx\_norm\_col\_tf\_idf.csv") %>%  
 rowwise() %>%  
 mutate(total=c\_across(Negative:Positive)%>%sum) %>%  
 mutate\_at(vars(Negative:Positive),~./total)

## Parsed with column specification:  
## cols(  
## sentiment = col\_character(),  
## Negative = col\_double(),  
## Neutral = col\_double(),  
## Positive = col\_double()  
## )

## # A tibble: 3 x 5  
## # Rowwise:   
## sentiment Negative Neutral Positive total  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 Negative 0.584 0.185 0.231 1059  
## 2 Neutral 0.421 0.354 0.225 1768  
## 3 Positive 0.220 0.188 0.592 1197

# (Successful) Attempt 2: Machine Learning

Bring sentiment column: allocate twice the weight to Negative which becomes the “TRUE”

df\_urdu\_dtm\_y <- df\_urdu\_dtm %>%  
 # double the weight on negatives  
 mutate(weight=if\_else(sentiment=="Negative",2,1)) %>%  
 select(sentiment,weight,everything()) %>%  
 # filter(sentiment!="Neutral") %>%  
 mutate(sentiment=sentiment=="Negative")

h2o.init()

## Connection successful!  
##   
## R is connected to the H2O cluster:   
## H2O cluster uptime: 21 hours 1 minutes   
## H2O cluster timezone: America/Sao\_Paulo   
## H2O data parsing timezone: UTC   
## H2O cluster version: 3.30.0.1   
## H2O cluster version age: 2 months and 28 days   
## H2O cluster name: H2O\_started\_from\_R\_drezn\_pia634   
## H2O cluster total nodes: 1   
## H2O cluster total memory: 3.36 GB   
## H2O cluster total cores: 4   
## H2O cluster allowed cores: 4   
## H2O cluster healthy: TRUE   
## H2O Connection ip: localhost   
## H2O Connection port: 54321   
## H2O Connection proxy: NA   
## H2O Internal Security: FALSE   
## H2O API Extensions: Amazon S3, Algos, AutoML, Core V3, TargetEncoder, Core V4   
## R Version: R version 4.0.2 (2020-06-22)

# h2o.shutdown()

XGBoost only available on mac or linux

h2o.xgboost.available()

Slow: do any of the columns have NA

any\_na <- function(vals) any(is.na(vals))  
  
df\_urdu\_mtx %>% as\_tibble %>%  
 mutate(has\_na=pmap\_lgl(.,~any\_na(list(...)))) %>%  
 filter(has\_na)

Convert to h2o matrix

Cast to h2o data frame

df\_urdu\_h2o <- df\_urdu\_dtm\_y %>%  
 as.h2o()  
df\_urdu\_h2o

AutoML main call (trains a bunch models, with 5-fold CV), 720s

ml\_urdu <- h2o::h2o.automl(y="sentiment",  
 weights\_column = "weight",  
 training\_frame=df\_urdu\_h2o,  
 max\_runtime\_secs = 720,  
 seed=0,  
 stopping\_metric = "misclassification")

Write leaderboard to file

ml\_urdu@leaderboard %>% as\_tibble %>% write\_csv("leaderboard.csv")

Recover leaderboard

df\_leaderboard <- read\_csv("leaderboard.csv")

## Parsed with column specification:  
## cols(  
## model\_id = col\_character(),  
## auc = col\_double(),  
## logloss = col\_double(),  
## aucpr = col\_double(),  
## mean\_per\_class\_error = col\_double(),  
## rmse = col\_double(),  
## mse = col\_double()  
## )

Show leader board, getting ~74% AUC

df\_leaderboard %>%  
 mutate(model\_id=str\_sub(model\_id,end=20)) # %>%

## # A tibble: 18 x 7  
## model\_id auc logloss aucpr mean\_per\_class\_error rmse mse  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 StackedEnsemble\_AllM 0.746 0.494 0.546 0.320 0.403 0.162  
## 2 StackedEnsemble\_Best 0.745 0.495 0.545 0.320 0.403 0.162  
## 3 GLM\_1\_AutoML\_2020063 0.739 0.587 0.677 0.364 0.449 0.201  
## 4 GBM\_grid\_\_1\_AutoML\_2 0.717 0.609 0.663 0.390 0.459 0.211  
## 5 GBM\_grid\_\_1\_AutoML\_2 0.712 0.615 0.658 0.399 0.462 0.214  
## 6 DRF\_1\_AutoML\_2020063 0.700 0.630 0.644 0.404 0.469 0.220  
## 7 GBM\_grid\_\_1\_AutoML\_2 0.700 0.618 0.648 0.403 0.463 0.215  
## 8 GBM\_grid\_\_1\_AutoML\_2 0.693 0.629 0.638 0.409 0.468 0.219  
## 9 GBM\_4\_AutoML\_2020063 0.669 0.640 0.621 0.433 0.474 0.224  
## 10 GBM\_3\_AutoML\_2020063 0.664 0.642 0.618 0.437 0.474 0.225  
## 11 GBM\_5\_AutoML\_2020063 0.661 0.646 0.593 0.443 0.477 0.227  
## 12 GBM\_1\_AutoML\_2020063 0.661 0.642 0.611 0.433 0.475 0.226  
## 13 GBM\_2\_AutoML\_2020063 0.660 0.643 0.612 0.432 0.475 0.226  
## 14 GBM\_grid\_\_1\_AutoML\_2 0.597 0.668 0.547 0.486 0.487 0.238  
## 15 XRT\_1\_AutoML\_2020063 0.541 0.680 0.468 0.495 0.494 0.244  
## 16 DeepLearning\_grid\_\_1 0.538 9.83 0.465 0.5 0.633 0.401  
## 17 DeepLearning\_grid\_\_2 0.530 4.91 0.472 0.5 0.620 0.385  
## 18 DeepLearning\_1\_AutoM 0.519 5.14 0.448 0.5 0.640 0.409

#knitr::kable() %>%  
 #kableExtra::kable\_styling(bootstrap\_options = c("striped", "hover", "condensed")  
#)

Save names of models and models to file (to be able to retrieve order)

ml\_urdu@leaderboard$model\_id %>%  
 as\_tibble %>%  
 write\_csv("saved\_models.csv")  
  
ml\_urdu@leaderboard$model\_id %>%  
 as.data.frame %>% pull(model\_id) %>%  
 head(6) %>%  
 walk(~h2o.saveModel(h2o.getModel(.x),path="models",force=T))

Read saved models list

df\_saved\_models <- read\_csv("saved\_models.csv")

## Parsed with column specification:  
## cols(  
## model\_id = col\_character()  
## )

Retrieve saved models as h2o model objects

list\_loaded\_models <- df\_saved\_models$model\_id %>%  
 head(6) %>%  
 map(~h2o.loadModel(str\_c("models/",.x)))

Report confusion matrix of the top model (~13% error rate for “Negative”=TRUE)

h2o.confusionMatrix(list\_loaded\_models[[1]])

## Confusion Matrix (vertical: actual; across: predicted) for max f1 @ threshold = 0.280899316839715:  
## FALSE TRUE Error Rate  
## FALSE 11202 3627 0.244588 =3627/14829  
## TRUE 1660 3627 0.313978 =1660/5287  
## Totals 12862 7254 0.262826 =5287/20116

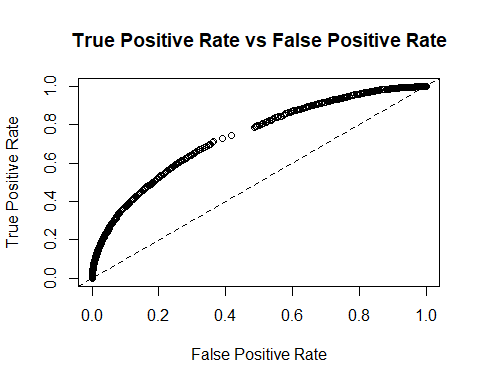
Plot AUC curve for 1st non-stacked model

model\_non\_stacked <- list\_loaded\_models%>%discard(~str\_starts(.x@model\_id,"Stacked"))%>%first  
model\_non\_stacked@model\_id

## [1] "GLM\_1\_AutoML\_20200630\_085522"

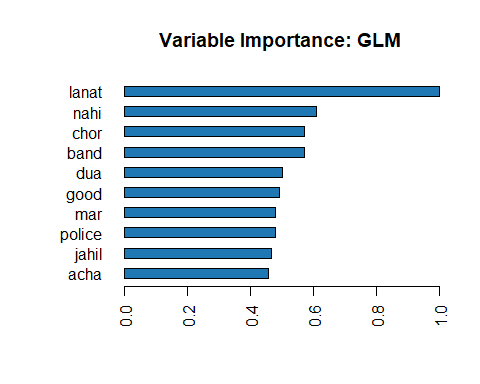
Plot AUC of top non-stacked model

plot(model\_non\_stacked%>%h2o.performance(xval=T),type="roc")



Plot variable importance for the first non-stacked model

h2o.varimp\_plot(model\_non\_stacked)



h2o::h2o.shutdown()

## Are you sure you want to shutdown the H2O instance running at http://localhost:54321/ (Y/N)?

# Business outcomes and Next Steps

## Business value

Assess individual or group sentiment of tweets, chats, etc. so as to: - Plan, train, alert customer care representatives - Fine-tune marketing messages which optimize sentiment - Quickly identify disgruntled employees, users, customers, bad brand interaction, etc.

## Suggested next steps

* Dataset
  + The sample is small (20k snippets), though the quality is generally good
* Potential fine-tunings to current approach
  + Play with sparsity threshold to include more or less words into vocabulary.
  + Use non-linear of dimensionality reduction, e.g. t-SNE, see picture below,
  + Test Union(Neutral,Positive) vs Negative
  + Use XGBoost, needs h2o on Linux or Mac
  + Could have separate classifier for “Neutral”
* Tried but ineffective
  + 3-class positive, negative, neutral (non-lumped)
  + PCA (no effective compression)
  + Play with TF\*IDF on document term matrix (initial tests were not good)
* Other approaches
  + Translate Urdu words into English and augment with English positive/negative sentiment word tables.
  + Augment the “bag of words” approach with order-depedency (use non-sparse 2- and 3-word sequences as features).
  + Use some order dependency to corroborate sentiment, LSTM or HMMs come to mind
  + Build correlation matrices of each of the non-sparse unique words and the positive/negative/neutral labels. Do a naive-bayes and/or perceptron classifier. Blend its predictions w/ the best ML one

## Types of dimensionality reduction

[source](https://cran.r-project.org/web/packages/dimRed/vignettes/dimensionality-reduction.pdf)

