Assignment: Business Intelligence Using Text Mining

Problem Statement:

A dataset of Shark Tank episodes contains 495 entrepreneurs making their pitch to the VC sharks.

1. A dataset of Shark Tank episodes contains 495 entrepreneurs making their pitch to the VC sharks.
2. You will ONLY use “Description” column for the initial text mining exercise.
3. Step 1:
    1. Extract the text into text corpus and perform following operations:
        a. Create Document Text Matrix
        b. Use “Deal” as a Dependent Variable
    2. Use CART model and arrive at your CART diagram
        1. Build Logistic Regression Model and find out your accuracy of the model
        2. Build randomForst model and arrive at your varImpPlot
4. Step 2:
    1. Now, add a variable to you analysis called as “ratio”. This variable is “askedfor/valuation”. (This variable is to be added as a column to your dataframe in Step 1.)
    2. Rebuild “New” models- CART, randomForest and Logistic Regression
5. Deliverables: (in a word document)
    1. CART Tree (Before and After) **25 Marks**
    2. RandomForest plot (Before and After) **25 Marks**
    3. Confusion Matrix of Logistic Regression (Before and After) **25 Marks**
    4. (Most important)- Your interpretation in plain simple English not extending more than half a page. **15 Marks**

SharkTank is a dataset of pitches done by entrepreneurs and founders to investors to get the investment for their business.

Here using Random forest, we will predict given the description of new pitch, how likely is the pitch will convert into success or not.

Import Dataset and Representation along with data cleaning

Import the shark tank dataset into R

```
#load the needed libraries

library(tm)

# Read in the data

data_sharktank = read.csv("Shark+Tank+Companies.csv", stringsAsFactors=FALSE)
```

Create Corpus by removing the noise.

```
# Create corpus
mycorpus = Corpus(VectorSource(data_sharktank$description))
myCorpus <- tm_map(myCorpus, tolower)
myCorpus <- tm_map(myCorpus,removePunctuation)
myCorpus <- tm_map(myCorpus, stripWhitespace)
```

#Document term matrix

```
dtm = DocumentTermMatrix(mycorpus)
inspect(dtm)
```

```
> inspect(dtm)
<<DocumentTermMatrix (documents: 495, terms: 5780)>>
Non-/sparse entries: 12893/2848207
Sparsity           : 100%
Maximal term length: 25
Weighting          : term frequency (tf)
Sample             :
     Terms
Docs  and are can for that the their with you your
  126   3   3   4   3    1   7     3    2   6    2
  179   5   2   0   1    0   2     2    3   5    1
  368   6   0   1   2    5   8     0    1   1    1
  379   3   0   0   1    1  10     2    0   1    2
  414   5   0   1   2    2   8     2    0   0    0
```

```
433    8   1   0   0    2  10      0    0   2    0
434    6   2   0   1    1   4      1    3   1    2
443    5   0   1   2    4  10      0    0   6    6
59     2   2   0   4    3  10      0    0   2    0
65     3   2   0   3    2   5      2    1   0    1
```

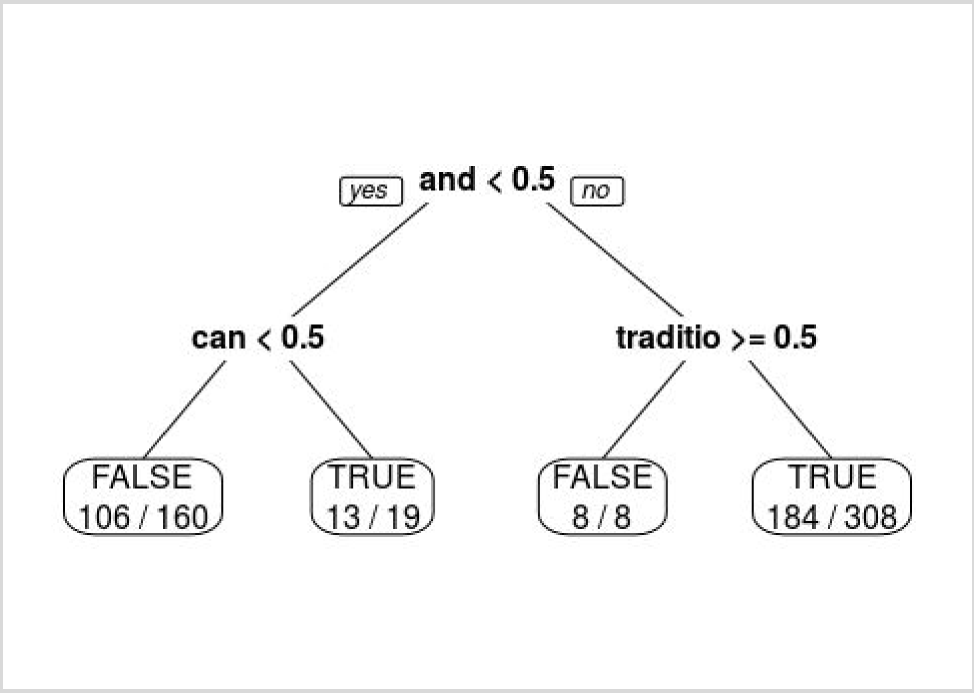
```
# Remove sparse terms whose frequency is less than 0.995
sparse = removeSparseTerms(dtm, 0.995)

# Convert to a data frame
descSparse = as.data.frame(as.matrix(sparse))

# Make all variable names R-friendly
colnames(descSparse) = make.names(colnames(descSparse),unique = TRUE)

# Add dependent variable Use “Deal” as a Dependent Variable
descSparse$deal = data_sharktank$deal

#Get no of deals
table(descSparse$deal)
```

```
> table(descSparse$deal)
FALSE  TRUE
  244   251
```

```
m = inspect(dtm)

# Convert to a data frame
DF = as.data.frame(m, stringsAsFactors = FALSE)

#write it to csv
write.csv(DF,"dtm.csv")
```

Build CART model by importing the needed libraries.

```
# Build CART model

library(rpart)
library(rpart.plot)

SharktankCart = rpart(deal ~ ., data=descSparse, method="class")

#CART Diagram
prp(SharktankCart, extra=2)
```

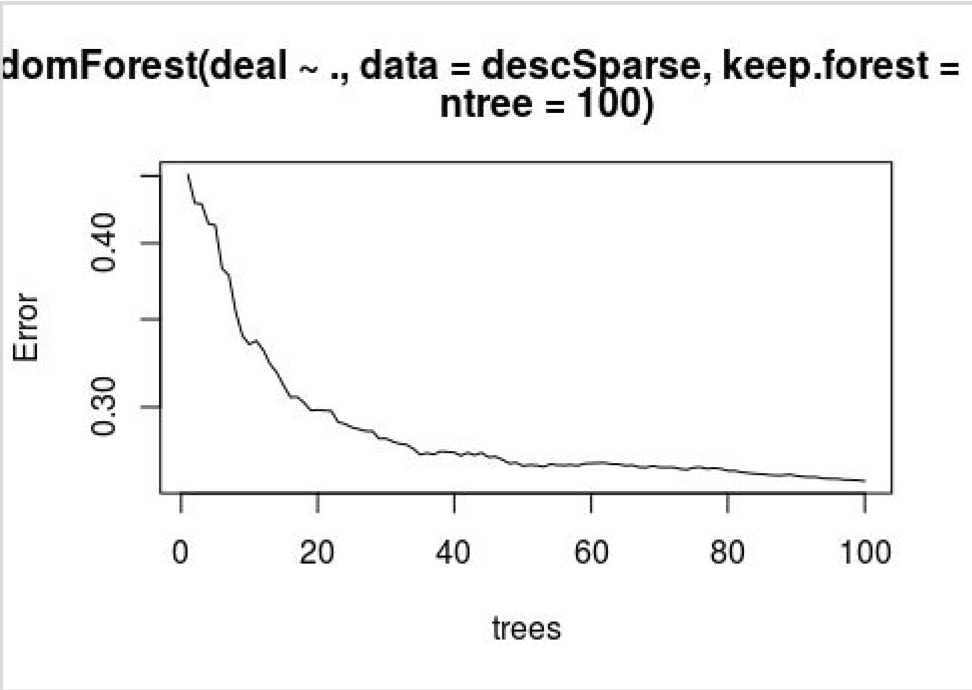


Evaluate the accuracy of CART model

```
# Evaluate the performance of the CART model
predictCART = predict(SharktankCart, data=descSparse, type="class")

CART_initial <- table(descSparse$deal, predictCART)

# Baseline accuracy
BaseAccCart = sum(diag(CART_initial))/sum(CART_initial)
BaseAccCart
```

```
> BaseAccuCart
[1] 0.6282828
```

```
# Random forest model
library(randomForest)
set.seed(123)

SharktankRF = randomForest(deal~., data=descSparse)
plot(randomForest(deal~., data=descSparse,keep.forest=FALSE, ntree=100), log="y")
```

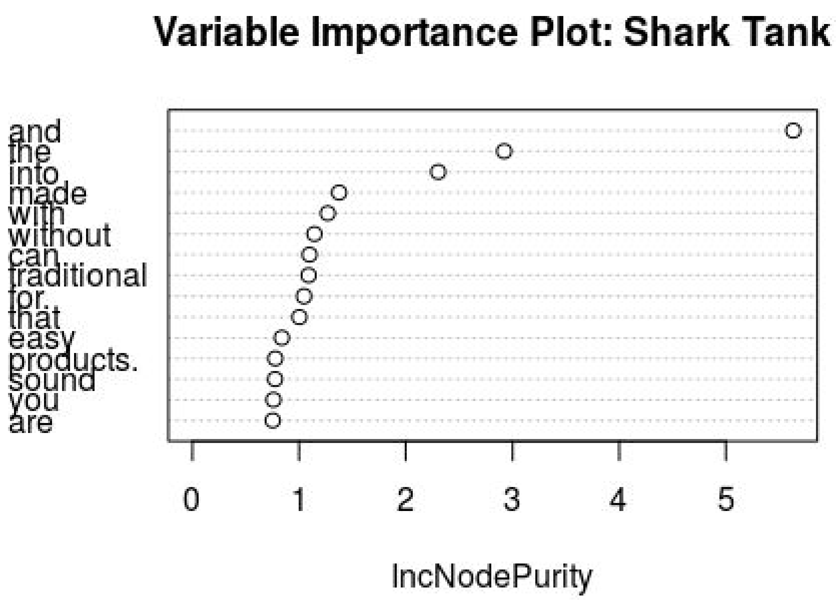



```
# Make predictions:
predictRF = predict(SharktankRF, data=descSparse)
```

# Evaluate the performance of the Random Forest
RandomForestInitial <- table(descSparse$deal, predictRF>= 0.5)

# Baseline accuracy
BaseAccRF = sum(diag(RandomForestInitial))/sum(RandomForestInitial)

#variable importance as measured by a Random Forest
varImpPlot(SharktankRF,main='Variable Importance Plot: Shark Tank',type=2)



#Logistic Regression model

set.seed(123)

Sharktanklogistic = glm(deal~., data = descSparse)

# Make predictions:
predictLogistic = predict(Sharktanklogistic, data=descSparse)

#confusion matrix
table(descSparse$deal, predictLogistic > 0.5)

```
> #confusion matrix
> table(descSparse$deal, predictLogistic > 0.5)

       FALSE TRUE
 FALSE   244    0
 TRUE      2  249
```

# Evaluate the performance of the Random Forest
LogisticInitial <- table(descSparse$deal, predictLogistic> 0.5)

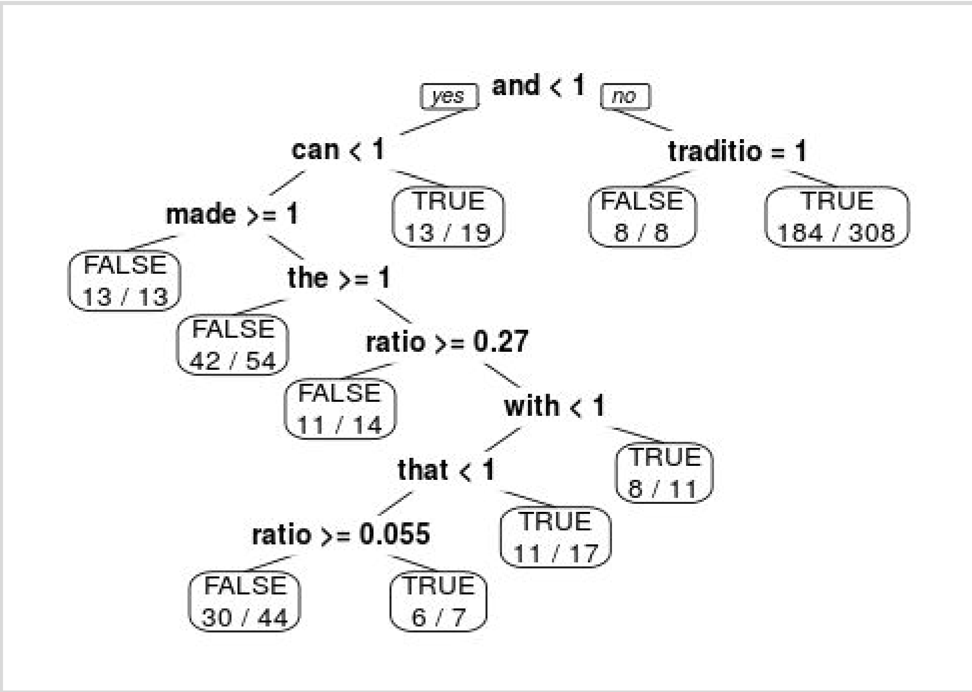
# Baseline accuracy
BaseAccuracyLogistic = sum(diag(LogisticInitial))/sum(LogisticInitial)

Add ratio variable into descSparse
descSparse$ratio = Sharktank$askedFor/Sharktank$valuation

#re-run the models to see if any changes

########CART Model###########
SharktankCartRatio = rpart(deal ~ ., data=descSparse, method="class")

#CART Diagram
prp(SharktankCartRatio, extra=2)

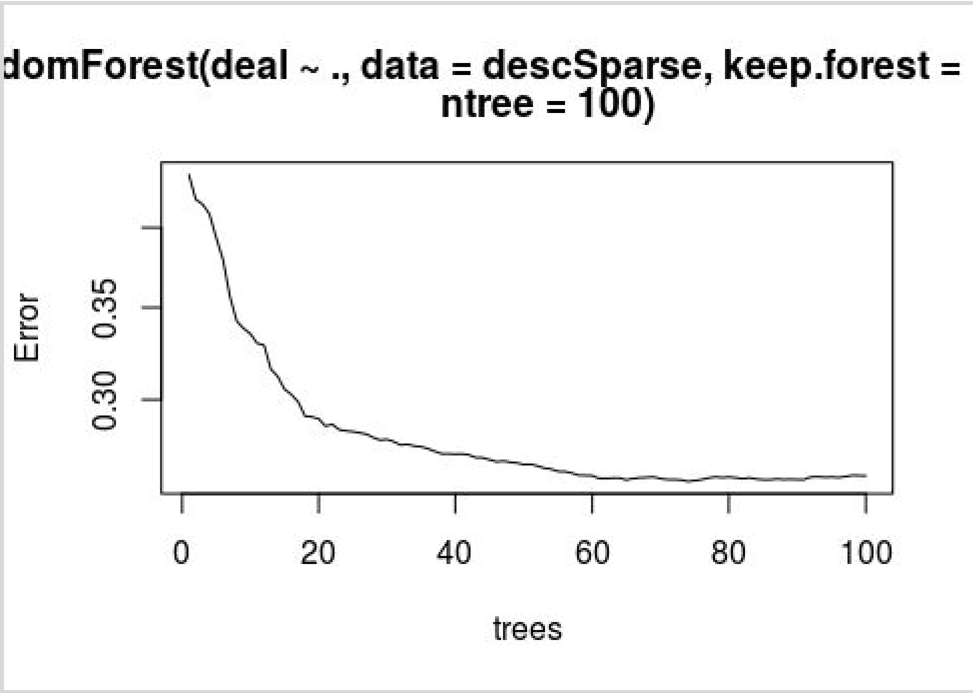


# Evaluate the performance of the CART model
predictCARTRatio = predict(SharktankCartRatio, data=descSparse, type="class")

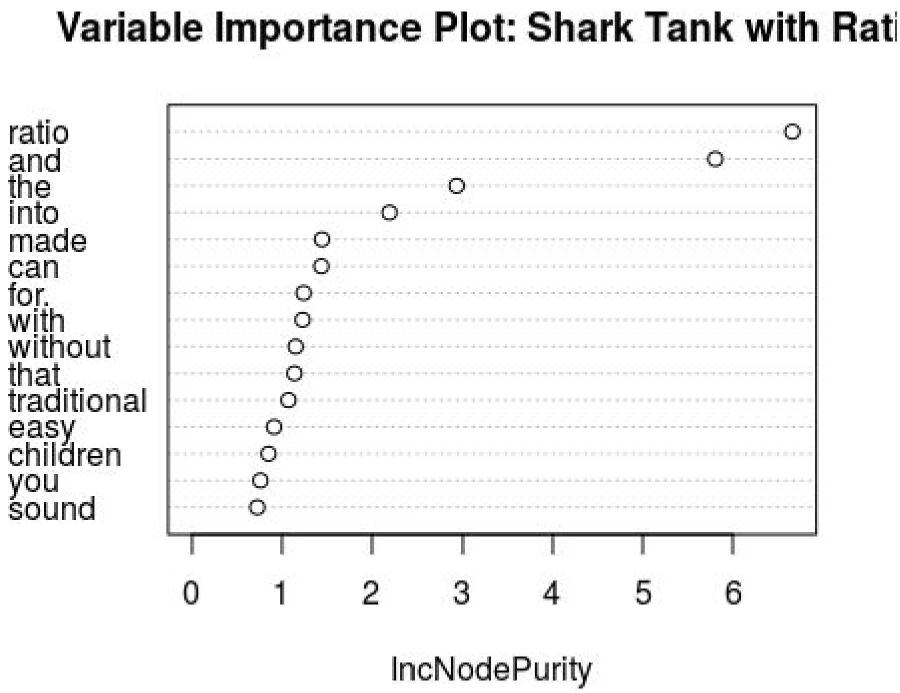
CART_ratio <- table(descSparse$deal, predictCARTRatio)

# Baseline accuracy
BaseAccuracyRatio = sum(diag(CART_ratio))/sum(CART_ratio)

```
#########Random Forest#############
#Random Forest Model
SharktankRFRatio = randomForest(deal ~ ., data=descSparse)
plot(randomForest(deal~., data=descSparse,keep.forest=FALSE, ntree=100), log="y")
```




```
#Make predictions:
predictRFRatio = predict(SharktankRFRatio, data=descSparse)

# Evaluate the performance of the Random Forest
RandomForestRatio <- table(descSparse$deal, predictRFRatio>= 0.5)

# Baseline accuracy
BaseAccuracyRFRatio = sum(diag(RandomForestRatio))/sum(RandomForestRatio)

#variable importance as measured by a Random Forest
varImpPlot(SharktankRFRatio,n.var=15,main='Variable Importance Plot: Shark Tank with
Ratio',type=2)
```




```
#########Logistic Regression##########
#Logistic Model
SharktanklogisticRatio = glm(deal~., data = descSparse)

# Make predictions:
predictLogisticRatio = predict(SharktanklogisticRatio, data=descSparse)

#confusion matrix
table(descSparse$deal, predictLogisticRatio > 0.5)
```

```
confusion matrix
> table(descSparse$deal, predictLogisticRatio > 0.5)

        FALSE  TRUE
  FALSE   244     0
  TRUE      0   251
```

```
# Evaluate the performance of the Random Forest
LogisticRatio <- table(descSparse$deal, predictLogisticRatio>= 0.5)

# Baseline accuracy
```

```
BaseAccuracyLogisticRatio = sum(diag(LogisticRatio))/sum(LogisticRatio)

####CART MODEL
#Before Ratio Column
BaseAccCart
> BaseAccCart
[1] 0.6282828

#After Ratio Column
BaseAccuracyRatio
> #After Ratio Column
> BaseAccuracyRatio
[1] 0.6585859

####RandomForest
#Before Ratio Column
BaseAccRF
> ####RandomForest
> #Before Ratio Column
> BaseAccRF
[1] 0.5555556

#After Ratio Column
BaseAccuracyRFRatio
> #After Ratio Column
> BaseAccuracyRFRatio
[1] 0.5636364


####Logistic Regression
#Before Ratio Column
BaseAccuracyLogistic
> ####Logistic Regression
> #Before Ratio Column
> BaseAccuracyLogistic
[1] 0.9959596


#After Ratio Column
BaseAccuracyLogisticRatio
> #After Ratio Column
> BaseAccuracyLogisticRatio
[1] 1
```