Rapport KAGGLE: Prédire si un post de blog va être populaire

Tout d'abord avant de commencer il fallait lire les fichier csv correspondant à la base d'apprentissage et à la base de test.

```
Train = read.csv("Train.csv", stringsAsFactors=FALSE)
Test = read.csv("Test.csv", stringsAsFactors=FALSE)
```

I) Remplissage des données manquantes

```
AllDatas <- rbind(Train[,-c(9,10,12)],Test)

AllDatas$NewsDesk[AllDatas$SectionName=="Opinion"] <- "OpEd"
AllDatas$NewsDesk[AllDatas$SectionName=="Style"] <- "Styles"
AllDatas$NewsDesk[AllDatas$SectionName=="Business Day"] <- "Business"
AllDatas$NewsDesk[AllDatas$SectionName=="Arts"] <- "Culture"
AllDatas$NewsDesk[AllDatas$SectionName=="Health"] <- "Science"
AllDatas$NewsDesk[AllDatas$SectionName=="Magazine"] <- "Magazine"
AllDatas$NewsDesk[AllDatas$SectionName=="N.Y. / Region"] <- "Metro"
AllDatas$NewsDesk[AllDatas$SectionName=="T:Style"] <- "TStyle"
AllDatas$NewsDesk[AllDatas$SectionName=="World"] <- "Foreign"
AllDatas$NewsDesk[AllDatas$SectionName=="Sports"] <- "Sports"
AllDatas$NewsDesk[AllDatas$SectionName=="Technology"] <- "Business"
AllDatas$NewsDesk[AllDatas$SectionName=="Crosswords/Games"] <-
"Business"
AllDatas$NewsDesk[AllDatas$SectionName=="Travel"] <- "Travel"
# AllDatas$NewsDesk[AllDatas$SectionName=="false" |
AllDatas$SubsectionName=="false"]<- "UnknownNewsDesk"


AllDatas$SubsectionName[AllDatas$SectionName=="Education"] <-
"Education"
AllDatas$SubsectionName[AllDatas$SectionName=="Multimedia"] <-
"Multimedia"
AllDatas$SubsectionName[AllDatas$SectionName=="Multimedia/Photos"] <-
"Multimedia/Photos"
# AllDatas$SubsectionName[AllDatas$SubsectionName=="false"] <-
"UnknownSubsectionName"
```

```
AllDatas$SectionName[AllDatas$SectionName=="Crosswords & Games"]  <-
"Crosswords/Games"
AllDatas$SectionName[AllDatas$NewsDesk=="Foreign"] <- "World"
AllDatas$SectionName[AllDatas$NewsDesk=="OpEd"] <- "Opinion"
# AllDatas$SectionName[AllDatas$SectionName=="false"] <-
"UnknownSectionName"

# AllDatas$NewsDesk[AllDatas$SubsectionName=="" &
AllDatas$SectionName==""] <- "UnknownNewsDesk"
# AllDatas$SectionName[AllDatas$SubsectionName=="" &
AllDatas$NewsDesk=="UnknownNewsDesk"] <- "UnknownSectionName"
# AllDatas$SubsectionName[AllDatas$NewsDesk=="UnknownNewsDesk" &
AllDatas$SectionName=="UnknownSectionName"] <- "UnknownSubsectionName"
```

En analysant les 2 dataset c'est à dire le fichier Train et le fichier Test j'ai remarqué qu'il y avait beaucoup de données manquantes dans NewsDesk , SectionName et SubsectionName que l'on pouvait remplir nous même avec une bonne observation. Tout d'abord il faut fusionner le Train et le Test pour pouvoir appliquer le remplissage des données manquantes sur les 2 bases. J'ai ainsi retiré les colonnes popular, Comments et Recommandations qui sont présentes uniquement dans le Train.
Le principe est le suivant par exemple dans l'échantillon de la dataframe AllDatas ci-dessous on remarque que certains NewsDesk ont été rempli pour le même SectionName.

	NewsDesk	SectionName
6	OpEd	Opinion
14		Opinion
17		Opinion
25	OpEd	Opinion
41		Opinion
45	OpEd	Opinion

En effet la SectionName "Opinion" a pour NewsDesk "OpEd" et ce sera toujours le cas il faut donc remplir tous les NewsDesk par Opinion. Il faut réaliser cela pour plusieurs cas. On réalise cela pour avoir plus de données et ainsi augmenter l'efficacité de notre algorithme.
J'ai également voulu remplir les lignes ne présentant aucune données par UnknownNewsDesk, UnknownSectionName et UnknownSubsectionName mais cela faisait baisser mon score du coup j'ai abandonné cette idée.

```
Train_bis <- head(AllDatas,nrow(Train))
Test <- tail(AllDatas, nrow(Test))
Train <- cbind(Train_bis,Train[,12])
colnames(Train)[11]="popular"

Train$Snippet <- NULL
Test$Snippet <- NULL
```

Ensuite j'ai réalisé un split de AllDatas après le travail de remplissage des données manquantes effectué pour pouvoir avoir de nouveau une dataframe Train_bis et une dataframe Test. De plus j'ai combiné la dataframe Train_bis avec la colonne popular de Train en une dataframe Train pour pouvoir réaliser la prédiction par la suite. J'ai également supprimé les colonnes Snippet dans Train et Test étant donné qu'ils sont similaire à la colonne Abstract présent dans ses 2 dataframe.

II) Features Engineering

Pour avoir la meilleur efficacité possible lors de l'apprentissage sur le Train et de la prédiction sur le Test il faut créer de nouvelles variables explicatives.

```
Train$PubDate = strptime(Train$PubDate, "%Y-%m-%d %H:%M:%S")
Test$PubDate = strptime(Test$PubDate, "%Y-%m-%d %H:%M:%S")

#Train$popular <- as.factor(Train$popular)
Train$Weekday <- Train$PubDate$wday
Train$hour <- Train$PubDate$hour
Train$monthday <- Train$PubDate$mday
Train$yearPubDate <- Train$PubDate$year
Train$month <- substr(Train$PubDate,6,7)
Test$Weekday <- Test$PubDate$wday
Test$hour <- Test$PubDate$hour
Test$monthday <- Test$PubDate$mday
Test$yearPubDate <- Test$PubDate$year
Test$month <- substr(Test$PubDate,6,7)
```

Tout d'abord il faut convertir la date en un format que R va comprendre. Ensuite pour créer nos nouvelles variable explicatives il faut extraire les informations de PubDate. Il faut que les variables explicatives créées soit présentes dans le Train mais aussi le Test. Grâce à PubDate on peut créer différentes variables. Tout d'abord une variable Weekday a donc été créé qui correspond au jour de la semaine où l'article a été publié. De plus une variable hour qui correspond à l'heure qui a été publié, une variable monthday qui correspond au jour du mois où l'article a été publié. Enfin une variable yearPubDate qui correspond à l'année où la variable a été créé et une variable monthday qui correspond au mois où l'article a été créé.

```
Train$NewsDesk <- as.factor(Train$NewsDesk)
Train$SectionName <- as.factor(Train$SectionName)
Train$SubsectionName <- as.factor(Train$SubsectionName)
Train$Weekday <- as.factor(Train$Weekday)
Train$hour <- as.factor(Train$hour)
Train$monthday <- as.factor(Train$monthday)
Train$yearPubDate <- as.factor(Train$year)
```

```
Train$logWordCount <- log(Train$WordCount+1)
Train$month <- as.factor(Train$month)


Test$SubsectionName[Test$SubsectionName == "Pro Football"] <- ""
Test$NewsDesk <- as.factor(Test$NewsDesk)
Test$SectionName <- as.factor(Test$SectionName)
Test$SubsectionName <- as.factor(Test$SubsectionName)
Test$Weekday <- as.factor(Test$Weekday)
Test$hour <- as.factor(Test$hour)
Test$monthday <- as.factor(Test$monthday)
Test$yearPubDate <- as.factor(Test$year)
Test$logWordCount <- log(Test$WordCount+1)
Test$month <- as.factor(Test$month)
```

J'ai ensuite convertir toutes les variables en variables catégorielles grace à la fonction as.factor() sauf pour WordCount et Nb_multimedia qui sont des entiers. J'ai créé une variable explicative représentant le logarithme de Wordcount car il y a beaucoup de variations entre la valeur de WordCount pour chaque article.

```
Train$PubDate <- NULL
Test$PubDate <- NULL

levels(Test$SectionName) <- levels(Train$SectionName)
```

Il faut ensuite supprimer la colonne PubDate dans le Train et le Test étant donné que l'on a extrait toutes les informations utiles.

La fonction levels permet d'avoir le même nombre de catégories dans le SectionName du Train et celui du Test.

```
AllCorpus <- rbind(Train[,4:5],Test[,4:5])

# count the 'c' character
count.c = function(col, c) {
  sapply(sapply(gregexpr(paste0("[",c,"]"), col),function(x) {
as.integer(x>=0) * length(x) } ), max)
}
```

Création d'une dataframe AllCorpus qui permet d'avoir les colonnes Headline et Abstract du Train et Test dans une même dataframe.

J'ai également créé une fonction count.c qui va permettre de compter le nombre d'un caractère que l'on veut chercher dans Headline et dans Abstract.

```
AllCorpus$nchar.head <- nchar(AllCorpus$Headline)
AllCorpus$nwords.head <- sapply(strsplit(AllCorpus$Headline, ' '),
length)
AllCorpus$nmaj.head <- count.c(AllCorpus$Headline, "A-Z")
AllCorpus$number.head <- count.c(AllCorpus$Headline, "0-9")
AllCorpus$dollar.head <- count.c(AllCorpus$Headline, "$")
AllCorpus$exclam.head <- count.c(AllCorpus$Headline, "!")
AllCorpus$quest.head <- count.c(AllCorpus$Headline, "?")
AllCorpus$quote.head <- count.c(AllCorpus$Headline, "'")
AllCorpus$deux_points.head <- count.c(AllCorpus$Headline, ":")
AllCorpus$logncharhead <- log(AllCorpus$nchar.head+1)

AllCorpus$nchar.abstract <- nchar(AllCorpus$Abstract)
AllCorpus$nwords.abstract <- sapply(strsplit(AllCorpus$Abstract, ' '),
length)
AllCorpus$nmaj.abstract <- count.c(AllCorpus$Abstract, "A-Z")
AllCorpus$number.abstract <- count.c(AllCorpus$Abstract, "0-9")
AllCorpus$dollar.abstract <- count.c(AllCorpus$Abstract, "$")
AllCorpus$exclam.abstract <- count.c(AllCorpus$Abstract, "!")
AllCorpus$quest.abstract <- count.c(AllCorpus$Abstract, "?")
AllCorpus$quote.abstract <- count.c(AllCorpus$Abstract, "'")
AllCorpus$deux_points.abstract <- count.c(AllCorpus$Abstract, ":")
AllCorpus$logncharabstract <- log(AllCorpus$nchar.abstract+1)
```

J'ai ensuite utilisé la fonction count.c pour compter par exemple le nombre de majuscule dans Headline et dans Abstract qui font référence à des mots important comme pays, ville, homme politique… J'ai également compté le nombre de points d'interrogations car j'ai supposé que les personnes seront attiré par des articles ayant comme Headline une question ou des questions présentes dans le Abstract.

```
AllCorpus$Headline <- gsub("New York|New York City", "NewYork",
AllCorpus$Headline, ignore.case = TRUE)
AllCorpus$nNewYork.head <- count.c(AllCorpus$Headline, "NewYork")
AllCorpus$Abstract <- gsub("New York|New York City", "NewYork",
AllCorpus$Abstract, ignore.case = TRUE)
AllCorpus$nNewYork.abstract <- count.c(AllCorpus$Abstract, "NewYork")
```

Ensuite j'ai collé le mot New et York dans le cas où New York ou New York City apparaît dans Headline ou bien abstract pour chaque article. c'est le principe de ngrams comme les deux mots ont une relation entre eux il ne faut pas fausser les résultats. De plus les articles proviennent du New York Times du coup il y a énormément de chances que ce mot apparaissent dans un article. C'est pour cela que j'ai créé 2 variables explicatives nNewYork.head et nNewYork.abstract qui correspond au nombre d'occurrences de NewYork dans Headline et Abstract pour chaque article.

```
AllCorpus$Abstract <- gsub(pattern="\\W",replace = "
",AllCorpus$Abstract)
AllCorpus$Abstract <- gsub(pattern="\\â",replace =
"",AllCorpus$Abstract)
AllCorpus$Abstract <- gsub(pattern = "\\b[A-z]\\b{1}",replace="
",AllCorpus$Abstract)
AllCorpus$Abstract <- gsub(pattern="\\d",replace = "
",AllCorpus$Abstract)
AllCorpus$Headline <- gsub(pattern="\\â",replace =
"",AllCorpus$Headline)
AllCorpus$Headline <- gsub(pattern = "\\b[A-z]\\b{1}",replace="
",AllCorpus$Headline)
AllCorpus$Headline <- gsub(pattern="\\d",replace = "
",AllCorpus$Headline)
AllCorpus$Headline <- gsub(pattern="\\W",replace = "
",AllCorpus$Headline)
```

Ensuite il faut réaliser divers regex grâce à la fonction gsub qui va permettre justement d'éliminer les éléments inutiles pour établir la matrice d'occurrences de mots par la suite.

Le premier gsub permet de remplacer plusieurs espaces et la ponctuation par un seul espace. Le 2ème permet de supprimer les “â” qui posaient problèmes. Le 3ème permet de supprimer les mots de taille un c'est à dire toutes les lettres de l'alphabet. Et enfin le 4ème permet de supprimer les chiffres.
Ces étapes seront utiles par la suite pour la création des 2 corpus.

```
countTrain <- head(AllCorpus, nrow(Train))
countTest <- tail(AllCorpus, nrow(Test))

countTrain$Headline <- NULL
countTrain$Abstract <- NULL
countTest$Abstract <- NULL
countTest$Headline <- NULL

Train <- cbind(Train,countTrain)
Test <- cbind(Test,countTest)
```

Il faut ensuite effectuer un split de AllCorpus en countTrain et countTest dans lesquels on supprime Headline et Abstract pour pouvoir ajouter les nouvelles variables explicatives créées à notre Train ainsi qu'à notre Test.

```
CorpusHeadline = Corpus(VectorSource(AllCorpus$Headline))
```

```
# You can go through all of the standard pre-processing steps like we
did in Unit 5:

CorpusHeadline = tm_map(CorpusHeadline, tolower)

CorpusHeadline = tm_map(CorpusHeadline, removePunctuation)

CorpusHeadline = tm_map(CorpusHeadline, removeWords,
stopwords("english"))

CorpusHeadline = tm_map(CorpusHeadline, stemDocument)


dtm = DocumentTermMatrix(CorpusHeadline)

sparse = removeSparseTerms(dtm, 0.99)

HeadlineWords = as.data.frame(as.matrix(sparse))

colnames(HeadlineWords) <- paste(colnames(HeadlineWords),"head",sep =
".")

colnames(HeadlineWords) = make.names(colnames(HeadlineWords))

HeadlineWordsTrain = head(HeadlineWords, nrow(Train))
HeadlineWordsTest = tail(HeadlineWords, nrow(Test))
TrainB = cbind(Train,HeadlineWordsTrain)
TestB = cbind(Test,HeadlineWordsTest)

TrainB$Headline <- NULL
TestB$Headline <- NULL
```

Le but ici est de créer la matrice d'occurrences des mots. Tout d'abord il faut créer un corpus de Headline. Ensuite convertir chaque mot en minuscule pour éviter les doublons, puis retiré la ponctuation. Après il faut également supprimer les stopwords qui sont des mots inutiles pour la prédiction de la popularité de l'article. Il faut ensuite stemmatiser le corpus. La stemmatisation est une sorte de normalisation car de nombreuses variantes de mots ont le même sens. La raison pour laquelle il est important d'effectuer la stemmatisation car cela permet de raccourcir la recherche et ainsi de normaliser les phrases.
Ensuite transformation du document en matrice de mots, ainsi que la transformation.
Ensuite transformation de la matrice d'occurrences des mots en dataframe appelé HeadlineWords. Il faut réaliser également un split de HeadlineWords en HeadlineWordsTrain et HeadlineWordsTest pour pouvoir ensuite ajouter la matrice des mots au Train et au Test. Enfin suppression des colonnes Headline dans Train et Test car toutes les données pertinentes ont été extraites.

III) Prédiction

```
set.seed(112)
forest <- randomForest(popular ~.-UniqueID-WordCount,data = TrainB,
                       nodesize = 2,na.action=na.exclude,ntree =
1000,importance = TRUE)

print(forest)
importance(forest)
varImpPlot(forest)

pred_forest_test <- predict(forest, TestB,type="response")

MySubmission = data.frame(UniqueID = TestB$UniqueID, Probability1 =
pred_forest_test)

write.csv(MySubmission, "Forest_1000_arbres_sans_abstractwords.csv",
row.names=FALSE)

trainPredforest <- predict(forest, type = "response")
table(TrainB$popular, trainPredforest > 0.5)
as.numeric(performance(prediction(trainPredforest, TrainB$popular),
"auc")@y.values)

forest$importance[order(forest$importance[, 1], decreasing = TRUE), ]
```

Tout d'abord il faut utiliser la fonction set.seed() dans laquelle on va mettre un chiffre entier pour que la prédiction affiche toujours les même résultats.

Pour mon modèle j'ai utilisé le randomForest qui m'a donné les meilleurs résultats. Tout d'abord il faut entraîner son modèle sur le Train en mettant comme formule popular~.-UniqueID-WordCount. en Effet UniqueID n'est pas pertinente étant donné que l'id d'un article n'aura pas d'influence sur la popularité. J'ai notamment exclus tous les NA et choisit 1000 arbres c'est avec cela que j'ai obtenu les meilleurs résultats.

Après il faut prédire sur le Test et enfin créé le fichier csv.

J'ai également calculé mon AUC sur la base d'apprentissage pour pouvoir connaître à peu près le score.