Predicting the Popularity of TED Talks

Aigerim Kulzhabayeva
Department of Mathematics and Statistics York University

Abstract In this report we will analyze audio-video recordings of TED Talks uploaded to the official TED.com website until September 21st 2017. We first develop a measure of popularity of the TED talk other than the typical measure of number of views and then build boosted tree model to predict the popularity of a TED Talk based on the available information. The resulting model has test AUC of 0.63 and test error of 0.07, both of which indicate that the model is not very good at predicting whether the video will be an outlier in terms of popularity or will fall in the common region.

Introduction

In this report we will analyze the data set that contain information about audio-video recordings of TED talks uploaded to the official TED.com website from 1984 - 2017. TED is a nonprofit organization devoted to spreading ideas usually in the form of short powerful talks given by experts in the field. TED talks cover topic ranging from science to business to global issues. The goal of the analysis is twofold. First, we note that the classical metric of number of views per video does not fully capture the popularity of the video. In this analysis we develop an alternative metric to measure the popularity of the video called the "Rating ratio" which is a ratio of the number of positive and negative ratings. Thus, we define ether video to be "popular" if it has an overwhelming number of positive ratings over negative rating,other wise the video is not popular. We create a binary variable of rating ratio where the variable takes value 1 if the video is an outlier in terms of rating ratio, and value 0 if the video has usual rating ratio. Second, once we have the popularity measure we try to predict the popularity of a TED Talk using the boosted trees method. The main questions to be addressed in this analysis are:

1. What are some other possibly better measures of popularity other than the standard measure of number of views?
2. What characteristics predict popularity of the TED Talks?

The following are the variables in the data set.

TABLE 1. *Description of the Data set*

Variable	*Description*
comments	The number of first level comments made on the talk
description	A description of what the talk is about
duration	The duration of the talk in seconds
event	The TED event where the talk took place
film date	The Unix timestamp of the filming
languages	The number of languages in which the talk is available
main speaker	The first named speaker of the talk
name	The official name of the TED Talk. Includes both the title and the speaker
num speaker	The number of speakers in the talk
published date	The Unix timestamp for the publication of the talk on TED.com
ratings	A string dictionary of the ratings given to the talk (e.g., inspiring, fascinating, jaw dropping, etc.) and their frequency
related talks	A list of dictionaries of recommended talks to watch next
speaker occupation	The occupation of the main speaker
tags	The themes associated with the talk
title	The title of the talk
url	The URL of the talk
views	The number of views on the talk.

Notes: Data from https://ssc.ca/en/case-study/case-study-2-what-predicts-popularity-ted-talks

In this analysis we understand the goal of identifying variables associate with the talk as variables which may be manipulated in the future with potential to affect the popularity of the talk. For this purpose we separate the variables temporally into three groups. The first group are the variables that affect the talk itself. Second group are the variables that are generated once the talk is videotaped. Last group are the variables that are collected once the talk is published on the internet.

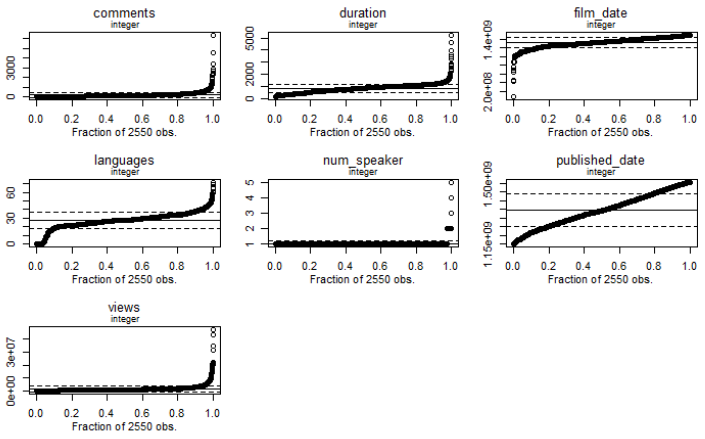
TABLE 2. *Temporal Order of Variables*

Prior to the Talk	**After the Talk**	**Online**
Duration	Tags	Number of Views
Event	Description	Number of Comments
Main Speaker	Published Date	Languages
Number of speakers	Related Talks	Ratings
Film Date		
Name		
Speaker Occupation		
Title		
URL		

The analysis would be different if we were purely looking at the variables that are just strongly associated with the predictor.

Data cleaning

We fist look at the data set using xqplot() function in spida2 package and identify two things: there is no missing data in the data set, and there are a lot of outliers in the variables "comments" and "views" visible in the plot below. However, instead of being nuisance, these clusters of outliers are exactly the extraordinary videos we would like to model.



Since the data set contain text variables which need to be processed individually in the following subsections we will proceed a few variables at the time.

Text Analysis in R

The TED Talks data set contains numerous string variables which contain information crucial for analysis. To deal with these variables we will be using regular expressions which is a programming tool available in most languages. Regular expression, also called Regex, is a filter that describes a set of strings that matches the pattern using a sequence of characters, meta-characters and operators. In a nutshell a regular expression is a set of symbols which is used to find certain patterns or sub strings in a given string, for example emails, SIN numbers, phone numbers etc. from a large unstructured text. Regular expressions is extremely powerful used for a variety of purposes such as feature extraction from text, string replacement and other string manipulations. One line of regex can easily replace several dozen lines of programming codes. In R the most popular functions for regular expressions are

1. gsub(pattern, replacement, x)
2. grep(pattern, x,)

We will illustrate the power of regular expressions on the following string variable using the function grep()

```
names<-c("Sarah", "John", "Sam", "Sammy","Jonathan")
```

```
    [1] "Sarah"    "John"     "Sam"      "Sammy"    "Jonathan"
```

Regex Operators

```
grep("^S", name, value = TRUE) -  matches any string that starts
                                  with an "S".
[1] "Sarah" "Sam"   "Sammy"



grep("n", name, value = TRUE) -  matches any string that ends with
                                 an "n".
[1] "John"     "Johnatan"



grep("^Sarah$", name, value = TRUE) -  matches the string exactly
                                       from  beginning to the end.
[1] "Sarah"



grep("Sa*", name, value = TRUE) -  matches any string that has "S"
                                   followed by ZERO or more "a"s.
[1] "Sarah" "Sam"   "Sammy"



grep("Sa+", name, value = TRUE) -  matches any string that has "S"
                                   followed by ONE or more "a"s.
[1] "Sarah" "Sam"   "Sammy"



grep("Jo[h|n]", name, value = TRUE) -  matches any string that has
                                      "Jo" followed by "h" or "n".
[1] "John"     "Johnatan"
```

There are many other features such as: character classes, bracket expressions and greedy and lazy matches, which can be used to construct Regex expressions to extract particular patterns or sub strings in a given string. We refer interested reader to the following website for a brief overview: https://medium.com/factory-mind/regex-tutorial-a-simple-cheatsheet-by-examples-649dc1c3f285

Removed variables

- "name" - includes the name of the speaker given in the variable "main speaker" and title of the talk given in the variable "title", thus a redundant variable.
- "URL" - contains unique URLs for each video, no additional information was extracted from this variable.

Days online

From the variable "date published", we can extract variable "dayson" which is a variable of number of day since the video has been published on the internet.

Comments and Views

The variable "comments" contains the number of comments the video has received, similarly the variable "views" contains the number of times the video has been watched. Since it is only logical that the longer the video has been online the more views and comment it has a chance to receive we will normalize these variables using the "dayson" variable, thus getting new variables: "mean.views" and "mean.comments".

Event

The variable "event" contains the name of the event at which the talk took place. There are two major types of talks, one is the signature TED Talk event, the other is privately held TEDx events organized by local organizations. As a consequence we decide to create a binary variable called "size.event" where first level corresponds to the signature TED event and second level corresponds to privately held event.

Gender of the speaker

From the variable "main Speaker", we can extract the gender of the speaker. We do this using the "GenderGuesser" package. GenderGuesser package uses the API provided by a company called Genderize, which uses a predictive model to assign gender based on the given name and the birth year of the person. However, not all names can be guessed and 58 names out of 2550 had to be imputed manually.

Repeat speaker

From the name of the speaker we notice that some speakers are invited to speak again. Thus, we create a binary variable "more talk" for the speakers that were invited again.

Tags

Finally, we look at the variable "tags", which is a very important variable because it contains a lot of information given in other variables. Below we give an example of a value in the variable:

```
[1] "['children', 'creativity', 'culture', 'dance', 'education',
'parenting', 'teaching']"
[2] "['alternative energy', 'cars', 'climate change', 'culture',
'environment', 'global issues', 'science', 'sustainability',
'technology']"
[3] "['computers', 'entertainment', 'interface design', 'media',
'music', 'performance', 'simplicity', 'software', 'technology']"
```

We see that "tags" includes the information about the speakers industry, occupation, the topic of the talk as well as other important keywords. Since the information about the occupation of the speaker becomes redundant, we remove the variable "Speaker Occupation" from the data set.

After the data is cleaned and the variables are processes, we end up with a set of new variables shown in the table below: In the next section we will look at potential

TABLE 4. *The New Variables in Temporal Order*

Prior to the Talk	After the Talk	Online
Duration	Tags (444 variables)	Mean Views
Event (Binary)	Days online	Mean Comments
Film Date		Languages
Number of speakers		Ratings Ratio
Gender		More Talk

measures of popularity and devise the response variable.

Measure of popularity

After working on predictors, we are moving on to the response variables. Popularity of the TED talk can be measured using different variables. In the data set we have 4 potential variables which can be use as a measure of the popularity of the talk.

1. Number of views, which we averaged over the number of days online.
2. Number of comments, also averaged over the number of days online.
3. Number of languages the talk has been translated into.
4. The ratings variable.

When it comes the choice of variable as the measure of popularity, the choice depends on how the term "popular" is defined. We define the term "popular" as "liked, admired, or enjoyed by many people or by a particular person or group". For this reason we will be focusing on the variables that reflects positive attitude towards the video as our measure of popularity. In particular, the rating variable is a string that contains 10 positive ratings and 4 negative ratings with relative counts, example of which is given below:

```
[1] "[{'id': 7, 'name': 'Funny', 'count': 19645}, {'id': 1,
'name':'Beautiful', 'count': 4573}, {'id': 9, 'name':
'Ingenious','count': 6073},{'id': 3, 'name': 'Courageous',
'count': 3253},{'id': 11, 'name':'Longwinded', 'count': 387},
{'id': 2, 'name':'Confusing', 'count': 242},{'id': 8, 'name':
'Informative', 'count': 7346}, {'id': 22, 'name':'Fascinating',
'count': 10581},{'id': 21, 'name': 'Unconvincing', 'count':300},
{'id': 24, 'name': 'Persuasive', 'count': 10704}, {'id': 23,
'name':'Jaw-dropping', 'count': 4439}, {'id': 25, 'name': 'OK',
'count': 1174},{'id': 26, 'name': 'Obnoxious', 'count': 209},
{'id': 10, 'name': 'Inspiring', 'count': 24924}]"
```

We will split each possible rating into a separate column with counts for each. To create a response variable for these we will create variable "total positive ratings" which will be the sum of positive ratings. We will also create variable "total negative ratings" and sum all the negative ratings. Finally, we create variable "rating ratio" which is the ratio of the positive ratings and the negative ratings. We create a binary variable of rating ratio where the variable takes value 1 if the video is an outlier in terms of rating ratio, and value 0 if the video has usual rating ratio.

Modelling

In this section we move on to modelling the popularity of TED talks using a machine learning model called Boosted Decision Trees. In a nutshell the tree-based methods are a predictive model that segments the predictor space into a number of simple regions. To predict the observation we take the mean or the mode of the training observations in the region to which it belongs. The method is called a tree because the splitting rules used to segment the predictor space can be summarized in a tree. In the next subsection we overview the basic decision tree models, which will be used as a foundation for the boosted trees discussed in subsequently.

Decision Trees

Example given in James et al. If we are trying to predict the baseball player's salary based on number of years and number of hits. First we split the variable years in to < 4.5 years and ≥ 4.5 years. The predicted salary for these players is given an the mean of all players in their split.

Interpretation: Years is the most important factor in determining Salary, and player with more experience earn higher salaries that the one with more.

How to build a tree:

1. Divide the predictor space of $X_1, ..., X_J$ into J distinct and non-overlapping regions. We can fit regions of any shape. The goal is to find such that $R_1, ..., R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{1}$$

where $\hat{y}_{R_j}$ is the average response for the training observations within the jth box. We do this by selecting a predictor X_j and a cut point s, such that for

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{2}$$

where

$$R_1(j,s) = \{X \mid X_j < s\} \text{ and } R_2(j,s) = \{X \mid X_j \geq s\} \tag{3}$$

2. For every observation that falls into the region R_j, we make the same prediction, which is a mean of the response values for he training observations of R_j.

Advantages:

- Very intuitive and easy to explain.
- Can be applied to predict both continuous response variables (Regression trees) and categorical response variables (Classification trees), since our response is continuous we focus on Regression trees.
- Tend to work well on unprocessed data.
- No data pre-possessing is not required.

Disadvantages:

- Tend to overfit.
- Have high variance, meaning small perturbations in the data change the model greatly.

The most common way to improve a decision tree is not to use a single tree but to have an ensemble of trees fitted to the data, and have a composite prediction. Next, we discuss one approach to improving predictions called Boosting.

Boosting

In Boosting, instead of growing a single tree we grow a sequence of trees, where each tree is grown using the residuals from the previous tree.

Let $b = 1, ..., B$ be the number of trees to be grown, the steps of growing boosted trees is as follows:

1. Fit a decision tree to data as done previously. Let the fitted function be $\hat{f}^b(x)$.

2. The next tree is obtained by updating $\hat{f}(x)$ by adding a shrunken version of a new tree:
$$\hat{f}(x) = \hat{f}(x) + \lambda \hat{f}^b(x)$$
This will add small increments to each value to have less residual.
3. We update the residuals,
$$r_i \leftarrow r_i \lambda \hat{f}^b(x_i)$$
4. The output is a boosted model
$$\hat{f}(x_i) = \sum_{b=1}^{B} \lambda \hat{f}^b(x_i)$$

XGBoost package

To implement the boosted trees model we will be using the `xgboost` package in R. XGBoost stands for *Extreme Gradient Boosting*. The package supports different objective functions and can be implemented for categorical, continuous and ordinal response variables. Even though the package is relatively easy to implement, some data processing is needed prior to modelling.

Data pre-processing

`Xgboost` only handles numeric matrices, which means that we can not have the data in typical data frame. This poses no problem for numeric variables in the data, however all categorical variables need to be first converted into dummy variables, and then change their class into numeric. After this step has been done, we need to change the data from the data frame into a matrix. There are three types of matrices accepted by `xgboost`, these are:

1. *Dense* matrices, which are the typical matrices in R, which can be obtained by using base function `as.matrix()`.
2. Second type are *sparse* matrices, which are matrices of class `dgCMatrix` obtained from package `Matrix`. These matrices are matrices with a large number of zeros, turning them into a `dgCMatrix` class allows the computer to not store zeros in memory, thus increasing efficiency.
3. The third class of matrices is a class specific to `xgboost` package, called `xgb.DMatrix`, which is list that contains both test and train data and can be used in the modelling function xgb.train to do simultaneous computations on both train and test data.

Training and testing the model

The the pre-processed data set we will be working on is called `dd.xgboost`. All factors in this data set are numeric dummy variables.

First we will be splitting our data set into train and test data sets.

```
sample<-sample.int(n = nrow(dd.xgboost),
size = floor(.75*nrow(dd.xgboost)), replace = F)

train.boost<-dd.xgboost[sample, ]
test.boost<-dd.xgboost[-sample, ]
```

In order to run the model, we need to have the predictors and response as separate objects. Thus, we extract the response variable for the data sets and creating separate vectors `train.label` and `test.label`.

```
train.label<-as.numeric(train.boost$rating.response)
train.boost$rating.response<-NULL

test.label<-as.numeric(test.boost$rating.response)After
test.boost$rating.response<-NULL
```

Next we will be creating turning both data sets into a `xgb.DMatrix` objects. Each object will contain both the data set of predictors and the corresponding response variable.

```
train.boost<-xgb.DMatrix(data=train.boost,label=train.label)
test.boost<-xgb.DMatrix(data=test.boost, label=test.label)
```

We can combine the training and testing `xgb.DMatrix` matrices, and will use the resulting object in the model.

```
train.test<-list(train=train.boost,test=test.boost)
```

The following is the best model:

```
model.1<-xgb.train(data = train.boost, nrounds = 40,
max.depth = 17, eta = 0.1, nthread = 2,
watchlist = train.test, eval.metric = "auc",
eval.metric = "error", objective = "binary:logistic")
```

Performing the 5-fold cross validation:

```
xgbcv<-xgb.cv(data=train.boost.DM, nrounds = 40,
nfold = 3, max.depth = 7, eta = 0.1, nthread = 2,
watchlist = watchlist, eval.metric = "auc",
eval.metric = "error", objective = "binary:logistic")
```

Unfortunately, we could not produce significant predictive results, the best results are presented in the following table:

TABLE 5. *Results*

Train AUC	**Train Error**	**Test AUC**	**Test Error**
0.93	0.06	0.63	0.07

Another nice feature of the `xgboost` package is that we can see which variables were important to build the model using `xgb.importance()` function.

Conclusion

In this analysis we have created a new measure of popularity which consisted of a ratio of total popular ratings over the total of negative ones. We have categorized the predictor into a binary variable, in which observation was level one if it was on outlier in terms of popularity, and level two if the observation was not an outlier. To predict the popularity of the video we have used the boosted decision tree models executed using the `xgboost` package. The resulting model has test AUC of 0.63 and test error of 0.07, both of which indicate that the model is not very good at predicting whether the video will be an outlier in terms of popularity or will fall in the common region.