Group 182: YouTube trending videos exploration and analysis

First Name	Last Name	Monday or Tuesday class	Share project with ITMD 525? (Y or N)
Arturo	Pavón Estradé	Tuesday	N
Raquel	Noblejas Sampedro	Tuesday	N

Table of Contents

1. Introduction

The main objective of this project is to analyze and try to model a dataset that contains data from YouTube. This data is stored on a csv file published on: https://www.kaggle.com/datasnaek/youtube-new/data. It contains various information about the top trending videos and their metadata, separated in different regions.

The motivation under choosing this dataset and application comes from all these ideas that trend to have no fundament or logic behind them on what makes a video trendy. We want to explore how the data is distributed among different categories and which factors are common between all the trending videos that we have information about. We find interesting the time line that a trending video will follow, from the days that it takes to become trend to how many days a video would stay on the top, based on the different categories we are given. Other possibility is to be able to predict a category of a video with the other variables, since it may be possible that videos from the same categories follow the same patterns (days until being trending, number of interactions such as views or likes, number of comments, etc.)

On this document we will show the process we have followed in order to obtain some answers about the questions raised above, and what are the conclusions we have reached and the future lines that we see for this dataset.

2. Data

As we have indicated in the previous section, we are going to use the dataset "Trending YouTube video statistics" available at Kaggle platform. Please note that this dataset is a new version of an old one created by the same author, so the collected data that we used for this project begins from 11/14/2017.

This dataset is formed by information about daily trending YouTube videos from different regions (USA, Great Britain, Germany, Canada, and France), with up to 200 listed videos per day. Each region data is stored in a separated .csv file, and a JSON file for retrieving the categories specific in that file, because the *category_id* field vary depending of the region.

Tags in the dataset:

- video_id: alphanumeric value unique for each video.
- trending_date: follows the format YY.DD.MM.
- title
- channel_title: title of the channel to which the video belongs.
- category_id: refers to different category depending on the region we are working.
- publish_time: date and hour of publication of the video, in GTM I format.
- tags: Separated by | character, [none] is displayed if there are no tags for the video.
- views: number of views at the moment of collection.
- likes: number of likes at the moment of collection.
- dislikes: number of likes at the moment of collection.
- comment_count: number of comments at the moment of collection.
- thumbnail_link: URL to the thumbnail image of the video.
- comments_disabled:if this is true, the comments are disabled for this video.
- ratings_disabled: if this is true, the ratings are disabled for this video.

- video_error_or_removed: if this is true, it means that the video was removed by the user, or some unspecified error occurred during collection of the data.
- description

3. Problems to be Solved

At the project proposal, we proposed to analyze the behavior of the data to see if the features have a different impact depending on the region we are working. After exploring the dataset more deeply, we have seen that for modelling purposes the differences between regions are not especially important, so in this project we only going to use the data available for the US region.

First, we are going to try to predict for how long the videos stay trending in YouTube, using the publishing date and the dates of trending for each video. We will study how much do the other features influence in this prediction, and if we are capable of say how many days a video from a given category remains up in the list.

Secondly, we will try to guess to which category a video belongs to according to the rest of the features. We can examine if all trending videos have common characteristics for each category and try to discover any pattern. The objective in this case would be to create a predictive model that, using the number of likes and comments, the number of views, and other features made up by ourselves, we could say to with category the video most probably belongs to.

4. Data Processing

We import our dataset and create a copy where there is only a unique row for every different video. We do this selection by filtering the dataset by the category ID, so we can assure that every video that takes place in our analysis is unique. We also assign the name of the category to its respective category ID by using the information provided in the JSON.

Then, we are going to create several new features of the dataset from the features we already have, such as DaysToTrend, DaysTrending, LikeIncrease, DislikeIncrease, CommentIncrease and viewIncrease since a video became trend. We also replace those fields that are NA with zeros in order to prevent future compiler errors.

```
#Average days in trend per video category adding to the whole dataset with repeated values daystotrend column
initialdate = ymd(substr(USvideos$publish_time, start = 1,stop = 10))
trendingdate = ydm(USvideos$trending_date)
USvideos[["DaysToTrend"]] <- trendingdate -initialdate
USvideos$DaysToTrend <- as.numeric(USvideos$DaysToTrend)


video_id = df1$video_id
daystrending <- numeric()
likesIncrease <- numeric()
dislikesIncrease <- numeric()
commentIncrease <- numeric()
viewIncrease <- numeric()

pos <- 0
```

```r
#We calculate how many days a video taking the highest daytotrend-lowestdaystotrend and increases since they become trend
for(position in 1:length(df1$video_id)){
  aux <- subset(USvideos, USvideos$video_id == video_id[position])
  if(length(aux$DaysToTrend) > 1){
    dt <- (aux$DaysToTrend[length(aux$DaysToTrend)] - aux$DaysToTrend[1])[1]
    il <- (aux$likes[length(aux$likes)] - aux$likes[1])[1]
    ic <- (aux$comment_count[length(aux$comment_count)] - aux$comment_count[1])[1]
    id <- (aux$dislikes[length(aux$dislikes)] - aux$dislikes[1])[1]
    iv <- (aux$views[length(aux$views)] - aux$views[1])[1]
  }else{
    dt <- 1
    il <- 0
    ic <- 0
    id <- 0
    iv <- 0
  }
  daystrending <- c(daystrending, dt)
  likesIncrease <- c(likesIncrease, il)
  dislikesIncrease <- c(dislikesIncrease, id)
  commentIncrease <- c(commentIncrease, ic)
  viewIncrease <- c(viewIncrease, iv)

}

df1[["DaysTrending"]] <- daystrending
df1[["LikeIncrease"]] <- likesIncrease
df1[["DislikeIncrease"]] <- dislikesIncrease
df1[["CommentIncrease"]] <- commentIncrease
df1[["ViewIncrease"]] <- viewIncrease

daysTrendingAvg <- numeric()
likesCategory <- numeric()
dislikesCategory <- numeric()
commentsCategory <- numeric()
viewsCategory <- numeric()
viAvg <- numeric()
liAvg <- numeric()
ciAvg <- numeric()
diAvg <- numeric()

for(element in 1:length(strcategories)){
  daysTrendingAvg <- c(daysTrendingAvg, sum(df1$DaysTrending[which(df1$category_id == element)])/sum(df1$category_id == element))
  likesCategory <- c(likesCategory, sum(df1$likes[which(df1$category_id == element)])/sum(df1$category_id == element))
  dislikesCategory <- c(dislikesCategory, sum(df1$dislikes[which(df1$category_id == element)])/sum(df1$category_id == element))
  commentsCategory <- c(commentsCategory, sum(df1$comment_count[which(df1$category_id == element)])/sum(df1$category_id == element))
  viewsCategory <- c(viewsCategory, sum(df1$views[which(df1$category_id == element)])/sum(df1$category_id == element))
  viAvg <- c(viAvg, sum(df1$ViewIncrease[which(df1$category_id == element)])/sum(df1$category_id == element))
  liAvg <- c(liAvg, sum(df1$LikeIncrease[which(df1$category_id == element)])/sum(df1$category_id == element))
  ciAvg <- c(ciAvg, sum(df1$CommentIncrease[which(df1$category_id == element)])/sum(df1$category_id == element))
  diAvg <- c(diAvg, sum(df1$DislikeIncrease[which(df1$category_id == element)])/sum(df1$category_id == element))
}

daysTrendingAvg[is.na(daysTrendingAvg)] <- 0
likesCategory[is.na(likesCategory)] <- 0
dislikesCategory[is.na(dislikesCategory)] <- 0
commentsCategory[is.na(commentsCategory)] <- 0
viewsCategory[is.na(viewsCategory)] <- 0
viAvg[is.na(viAvg)] <- 0
ciAvg[is.na(ciAvg)] <- 0
diAvg[is.na(diAvg)] <- 0
liAvg[is.na(liAvg)] <- 0


daysTrendingAvg <- daysTrendingAvg[daysTrendingAvg>0]
likesCategory <- likesCategory[likesCategory>0]
dislikesCategory <- dislikesCategory[dislikesCategory>0]
commentsCategory <- commentsCategory[commentsCategory>0]
viewsCategory <- viewsCategory[viewsCategory>0]
viAvg <- viAvg[viAvg>0]
ciAvg <- ciAvg[ciAvg>0]
diAvg <- diAvg[diAvg>0]
liAvg <- liAvg[liAvg>0]
```

For the classification modelling, we divided the dataset in a training dataset and a testing dataset before we normalized the data.

```
#Dataset with numerical variables and unique id
#First we are going to take only the numerical variables from the original unique dataset(category_id, views,likes,dislikes,comment_count)
df2 <- df1[c('likes','views')]
ind <- sample(2, nrow(df2), replace=TRUE, prob=c(0.67, 0.33))
vid_train <- df2[ind==1,]
vid_test <- df2[ind==2,]

#Supervised learning, we cannot make the model with the labels column
train_lab <- factor(df1[ind==1,5])
test_lab <- factor(df1[ind==2,5])

#Normalize data
fnormalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
vid_norm_tr <-as.data.frame(fnormalize(vid_train))
vid_norm_ts <-as.data.frame(fnormalize(vid_test))
```

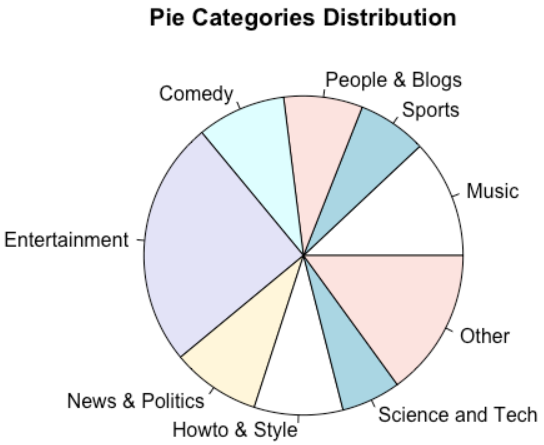
5. Methods and Process

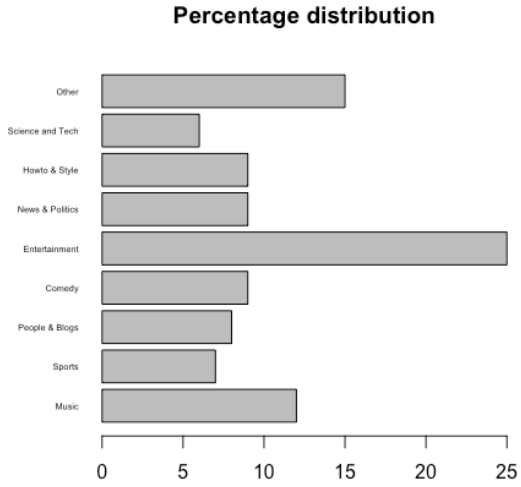
5.1. Statistics

In order to provide reasonable conclusions from our modelling, first we are going to try to understand what the behavior of the data is. In this phase of analysis, we removed all those outliers that we believe did not represent the true pattern and can distortion the result plots. This previous step, although it is not directly related with the main objectives of the project would save us time and give us clues about what are the features of the dataset that really have a meaningful impact over our target variables.

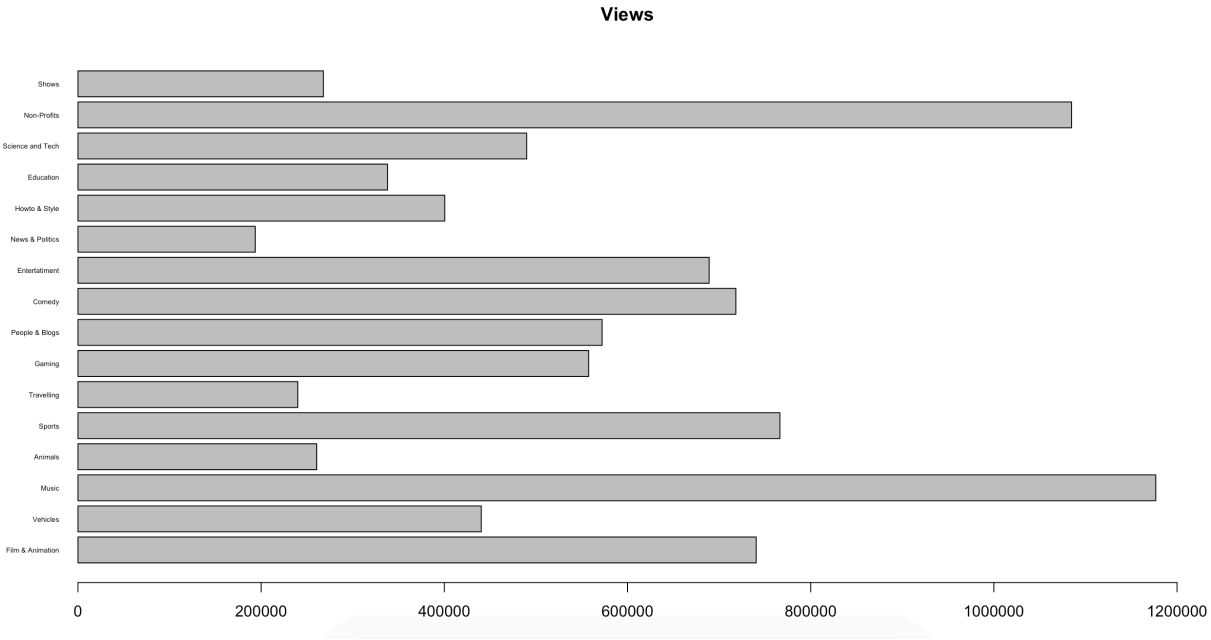
The first graph we generated expose the percentage distribution of the videos per category. Those categories that are below the five percent of the whole dataset are grouped together at a category named other. As we can see the more trending videos are those that belong to the Music category and the Entertainment category; this would make sense because of the apogee of the youtubers and people that publish content for a way of living.

```
#Calculation of percentages of each category
pct <- round((c/sum(c))*100)

#We obtain the position of our categories with at least 1 video and at least 5%
positions <- character()
for (element in 1:length(pct)){
  if (pct[element] > 5){
    positions <- c(positions, strcategories[element])
  }
}
pct <- pct[pct>5]

#Addition as last element of sum of the ones with less than 5% and give and id of 3
pct <- c(pct, 100-sum(pct))
positions <- c(positions, "Other")

#Plot of the percentage distribution
barplot(pct, names.arg = positions, las = 1, horiz = TRUE, xlim = c(0, 25), main = "Percentage distribution", cex.names = 0.45)
pie(pct, label = positions, radius = 0.9, main = "Pie Categories Distribution")
```





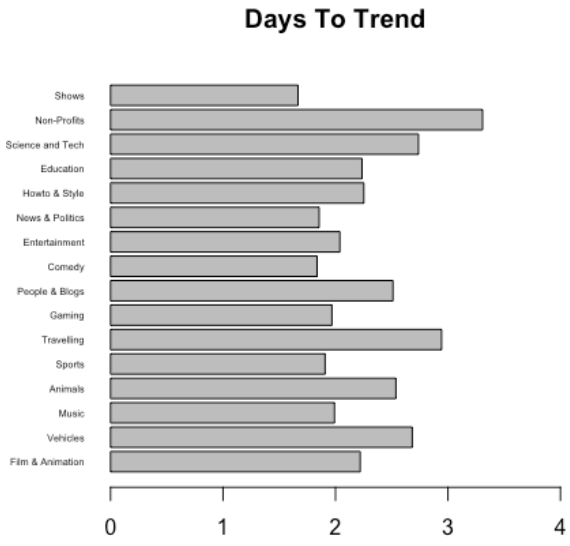
We also represented a graph with the average views per category, and it is interesting that the most viewed categories are not exactly the same as the categories that appears most on the trending list. The Non-profits value is that high because not long time ago there was a viral video that belonged to that category and the number of views went up.

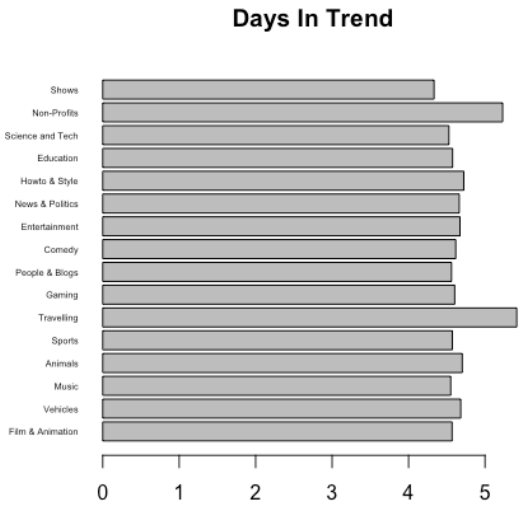


For the time series analysis, we thought it would be interesting to know how many days a video from a certain category needs for become trend, and also how many days in average a video stays in the top per category. Surprisingly, the category for this last plot seems to be really significant, because the average is about four days for all kind of videos.

```
#Calculation of  days to be trendy
initialdate = ymd(substr(df1$publish_time, start = 1,stop = 10))
trendingdate = ydm(df1$trending_date)
df1[["DaysToTrend"]] <- trendingdate -initialdate
df1$DaysToTrend <- as.numeric(df1$DaysToTrend)

df1 = subset(df1, df1$DaysToTrend<20) #-> Videos with relevant points

#Calculation of average day per category to be trendy
avgDayCat <- numeric()
for(element in 1:length(strcategories)){
  avgDayCat <- c(avgDayCat, sum(df1$DaysToTrend[which(df1$category_id == element)])/sum(df1$category_id == element))
}

avgDayCat[is.na(avgDayCat)] <- 0
dayPosition <- character()
for(element in 1:length(strcategories)){
  if(avgDayCat[element] != 0)
    dayPosition <- c(dayPosition, strcategories[element])
}
avgDayCat <- avgDayCat[avgDayCat>0]

barplot(avgDayCat, names.arg = dayPosition, las = 1, horiz = TRUE, xlim = c(0, 4), main = "Days To Trend", cex.names = 0.45)
```

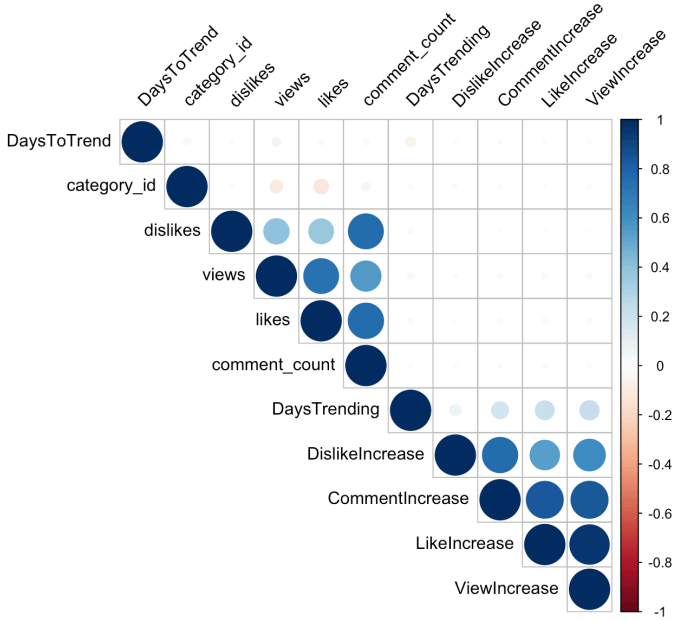
```
#Plot of average days in trend per category
barplot(daysTrendingAvg, names.arg = dayPosition, las = 1, horiz = TRUE, xlim = c(0, 5.5), main = "Days In Trend", cex.names = 0.45)
```

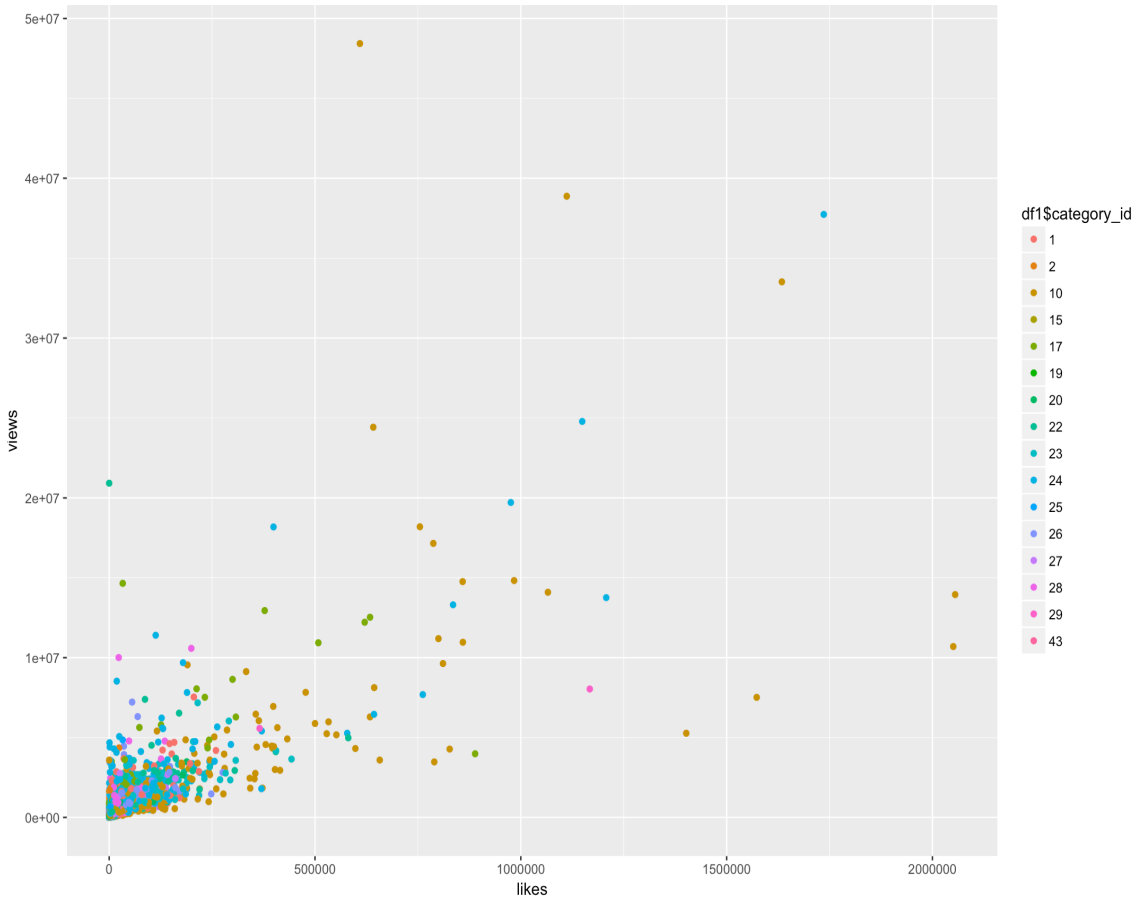




Lastly, for knowing which of the features would give us a good accuracy for the classification purposes, we wanted to see the distribution of the videos per category and also the correlation matrix of the dataset.

```
#Analyze correlation
df3 <- df1[c('views','likes','dislikes','comment_count','DaysToTrend','DaysTrending','LikeIncrease','DislikeIncrease','CommentIncrease','ViewIncrease','category_id')]
df3 <- sapply(df3, as.numeric )
res <- cor(df3)
corrplot(res, type = "upper", order = "hclust",tl.col = "black", tl.srt = 45)
#Draw clusters
df1$category_id <-as.factor(df1$category_id)
ggplot(df1, aes(views,likes, color = df1$category_id)) + geom_point()
```


DaysToTrend
category_id
dislikes
views
likes
comment_count
DaysTrending
DislikeIncrease
CommentIncrease
LikeIncrease
ViewIncrease
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


5e+07
4e+07
3e+07
2e+07
1e+07
0e+00
views
0
500000
1000000
1500000
2000000
likes
df1$category_id
1
2
10
15
17
19
20
22
23
24
25
26
27
28
29
43

If we take a look at these two graphs, we would conclude that it is going to be difficult to build a classifier because of the sample distribution, and also that the features that have a significant impact on the category ID variable are the likes and the views, or the likes increase and the views increase.

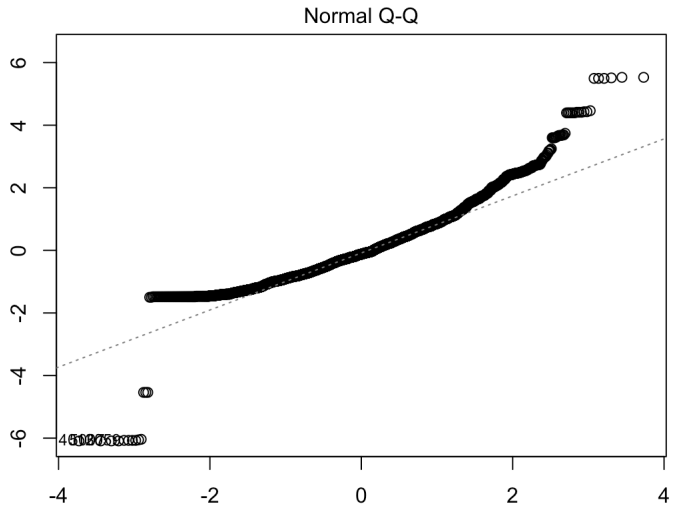
5.2. Linear regression model

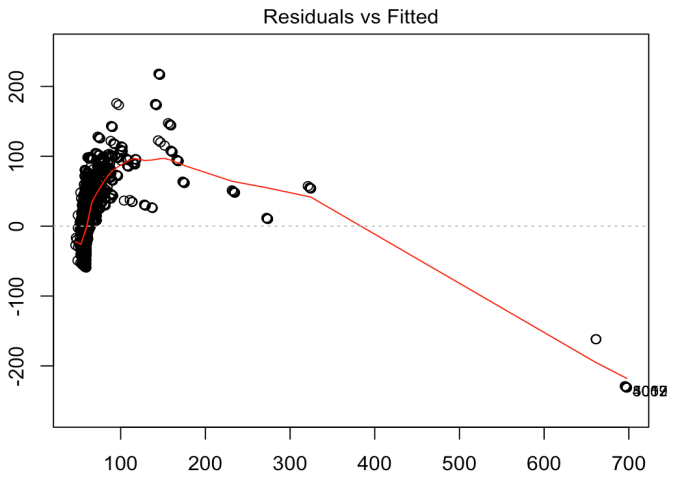
After trying many different models, we saw that the best transformation we could apply to this dataset would be the cubic root for the target variable "Views increase", and the model would be the following:

```
lm = lm((ViewIncrease)^(1/3) ~ CommentIncrease + LikeIncrease + DaysToTrend, data=df1)
plot(lm)
plot(resid(lm))
```

$$\sqrt[3]{VI} = 0.0002024CI + 0.0000274LI - 0.6642DTT + 0.5917$$

Although the application of the cube root transformation seems to linearize the model and the sample looks approximately normal, we can see that there is still a pattern in the residuals plot, so we conclude that this linear model cannot be used for this data set.





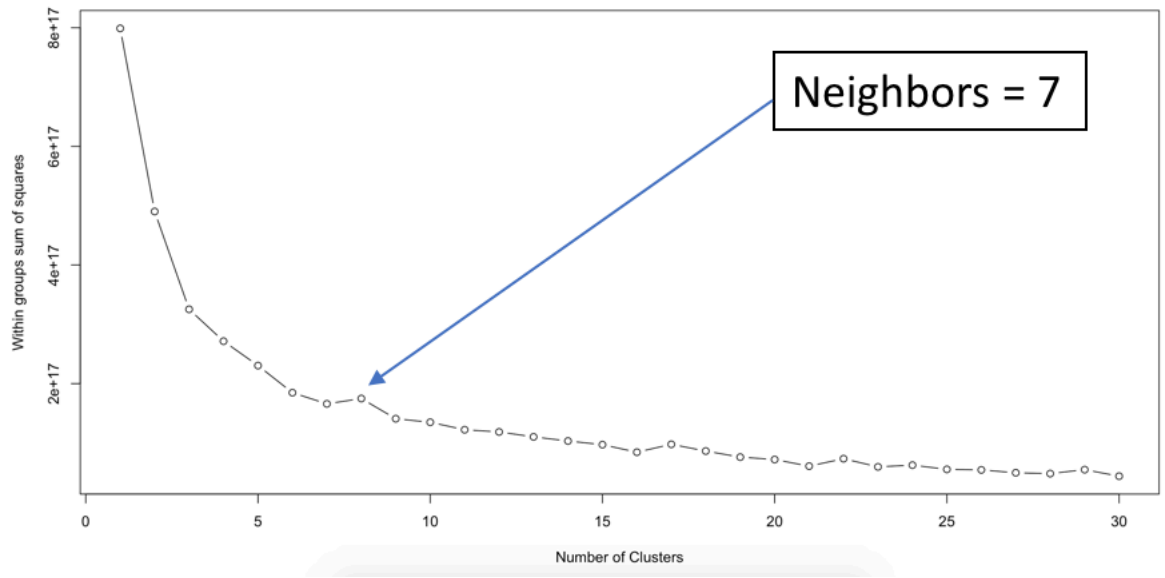
5.3. Classification

We have tried three different types of classifiers for this dataset, each of one with and without cross-fold validation in order to improve the model performance. As we have mentioned before the features with more impact over the category ID variable are the views and the likes, so we are going to use those variables to build our different classifiers.

The first classifier we have tried is the K-Nearest Neighbors; for non-cross validation we figured out that the best k value was 27 neighbors. For the cross-validation approach we used the Elbows method for choosing our k optimal value, which turned out to be 7.

```
###### KNN modeling

#Elbow method
wss <- (nrow(dfcat)-1)*sum(apply(dfcat,2,var))
for (i in 2:30) {
  wss[i] <- sum(kmeans(dfcat,centers=i)$withinss)
}
plot(1:30, wss, type="b", xlab="Number of Clusters",ylab="Within groups sum of squares")

#The best k executing is 27
knn <- knn(train=vid_norm_tr, test=vid_norm_ts, cl=train_lab, k=27)
```

```
#Cross-fold validation with KNN
knn2 =train(vid_norm_tr,train_lab,'knn',trControl=trainControl(method='cv',number=10))
knn2_pred <- predict(knn2,vid_norm_ts)
```



The second classifier we have used for predicting the category ID of a trending video is the Naïve Bayes classifier, also using cross-fold validation and not.

```
##### Naive Bayes

#Cross-fold validation with NB
nb2 =train(vid_norm_tr,train_lab,'nb',trControl=trainControl(method='cv',number=10))
nb2_pred<-predict(nb2$finalModel,vid_norm_ts)
#Accuracy
100*sum(nb2_pred$class == test_lab)/length(test_lab)

#Without cross-fold validation
vid_train <- sapply(vid_train, as.numeric )
vid_test <- sapply(vid_test, as.numeric )
nb = naiveBayes(vid_norm_tr,train_lab,laplace=1)
nb_pred = predict(nb,vid_norm_ts)
#Accuracy
100*sum(nb_pred == test_lab)/length(test_lab)
```

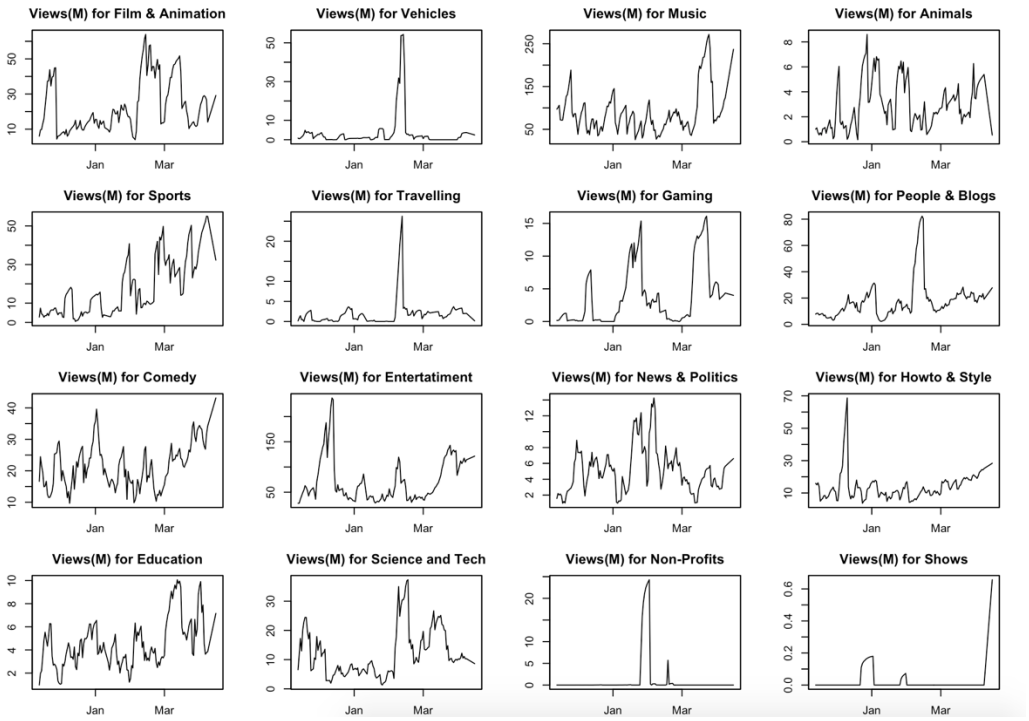
Finally, we built a decision tree with both variables to see if we could boost the classifier performance:

```
#Decision tree
vid_norm_tr[["category_id"]] <- train_lab
tree<-rpart(category_id~views+likes,data=vid_norm_tr)
tree_pred<-predict(tree,vid_norm_ts,type="class")
```

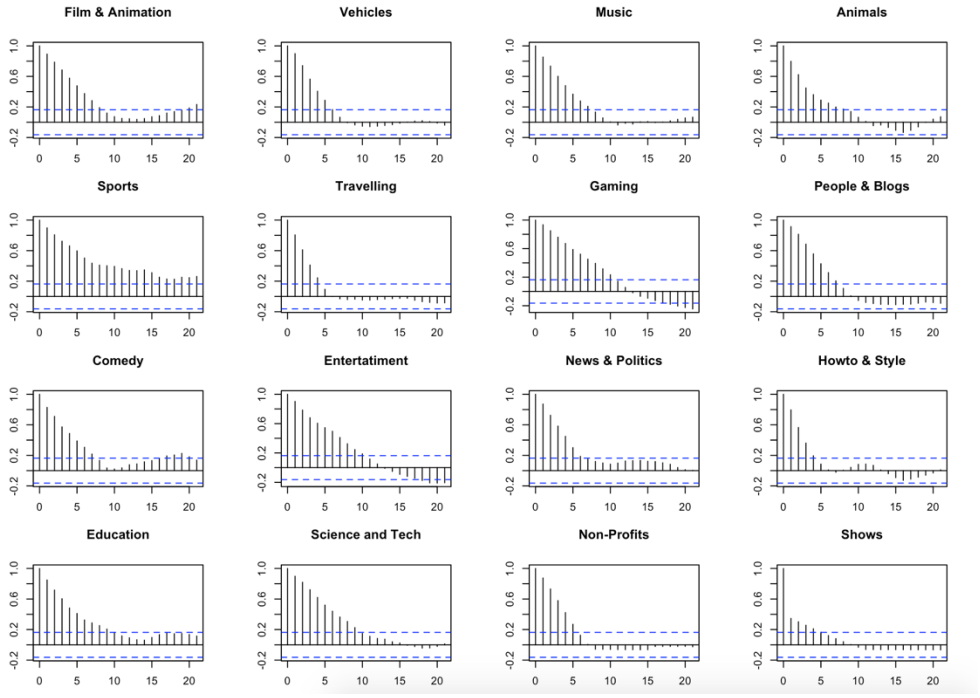
5.4. Time-series

At first, the initial idea could have been predicting the statistics from a video. The number of reproductions that it will have the following day, the number of likes/dislikes or comments. If we have a look at the statistics section, it is reasonable to discard this idea with the information that we have available. Since videos stays in trend on average for around four days, we will only have 4 points on average to do predictions on the videos. The times-series model that will result out of it would not be pretty accurate since the number of points is small.

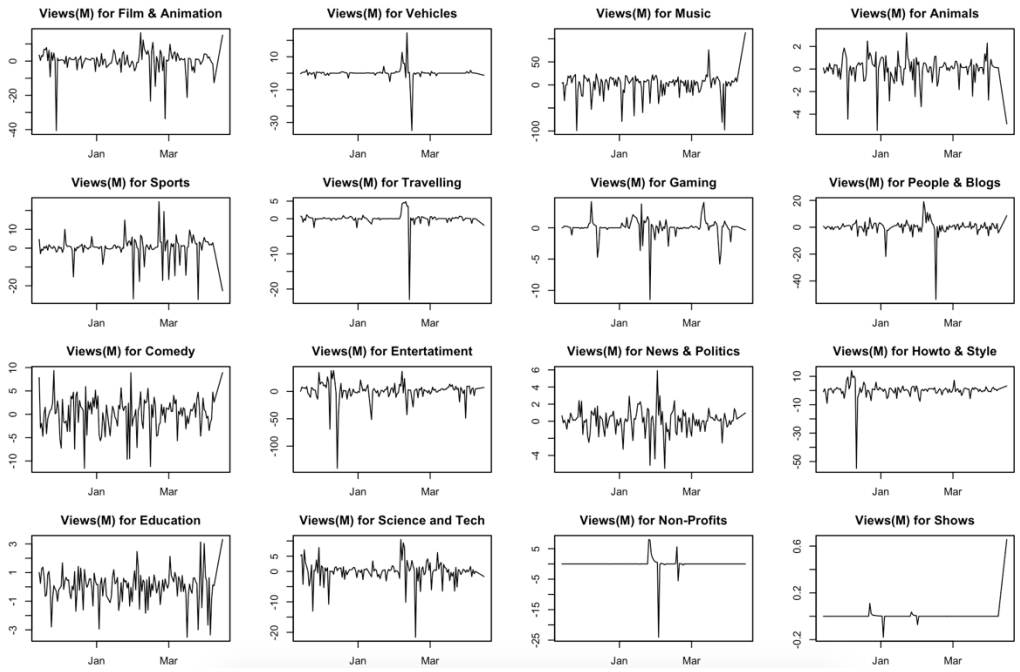
As a consequence, we decided to do the predictions out of the sum of all the different categories we have. For this case, we are going to analyze the behavior of the views per category using them as a starting point.



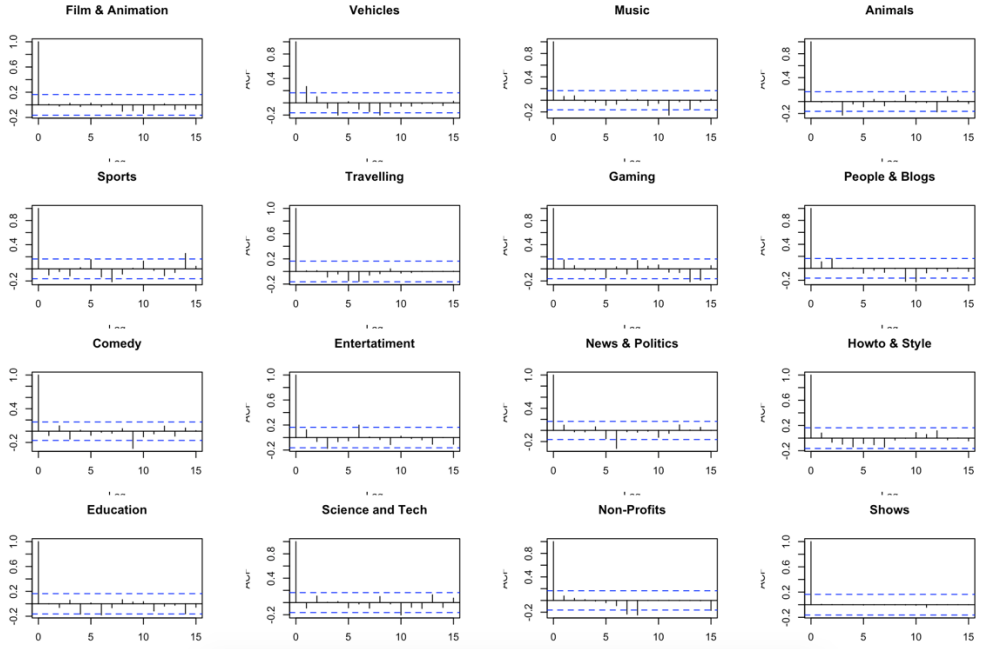
At a simple glance, it is likely to predict that there are no stationary behaviors from any of the categories. We can confirm that in our auto correlation plot for all the different categories.



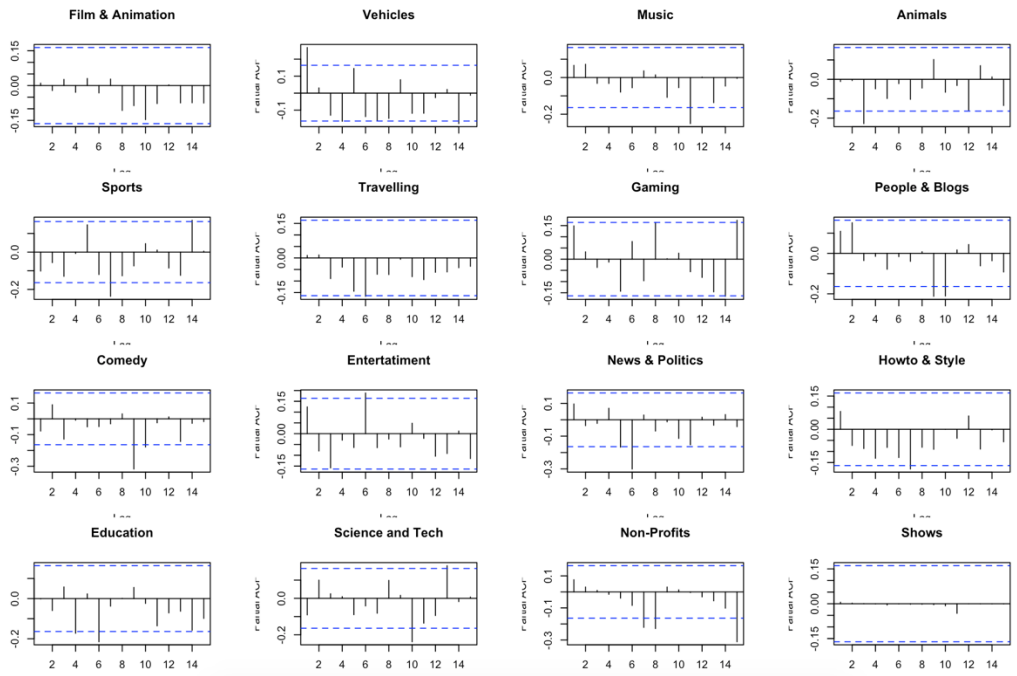
We can conclude that none of the categories are stationary because none of them decay fast enough to be considered stationary. To solve this problem, we are going to differentiate all the categories to see whether it is going to be possible to do a proper analysis using the views or not. These are the results of the evolution in our differentiation process for every category.



We can observe that now everything has been smoothed. It is time to observe the correlation and partial autocorrelation plots to see if we can create any model with the first differentiation of our data.



It is possible to observe that now our data is stationary since now the data is stationary. If we observe the partial autocorrelation plot, the behavior is distinct from what we have studied in class since not all the plots show relevant samples.



We can observe that only vehicles show the behavior we studied in class.

Regarding the serial correlation, we can compute Ljung Box test that shows that all the series are correlated with the exception of shows.

```
> L1

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 425.29, df = 20, p-value < 2.2e-16

> L2

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 292, df = 20, p-value < 2.2e-16

> L3

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 319.41, df = 20, p-value < 2.2e-16

> L4

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 246.66, df = 20, p-value < 2.2e-16

> L5

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 698.45, df = 20, p-value < 2.2e-16
```

```
> L6

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 192.05, df = 20, p-value < 2.2e-16

> L7

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 597.84, df = 20, p-value < 2.2e-16

> L8

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 403.7, df = 20, p-value < 2.2e-16

> L9

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 346.99, df = 20, p-value < 2.2e-16

> L10

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 505.51, df = 20, p-value < 2.2e-16

> L11

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 316.36, df = 20, p-value < 2.2e-16

> L12

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 178.67, df = 20, p-value < 2.2e-16

> L13

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 371.02, df = 20, p-value < 2.2e-16

> L14

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 476.37, df = 20, p-value < 2.2e-16

> L15

        Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 287.54, df = 20, p-value < 2.2e-16

> L16

       Box-Ljung test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 62.287, df = 20, p-value = 3.134e-06
```

To test the normality of the distribution we can observe that the only categories in which we cannot regret a normality distribution are vehicles, travelling , people & blogs and entertainment follow a normal distribution.

```
> J1

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 19.099, df = 2, p-value = 7.122e-05

> J2

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 2718.8, df = 2, p-value < 2.2e-16

> J3

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 110.87, df = 2, p-value < 2.2e-16

> J4

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 11.907, df = 2, p-value = 0.002597

> J5

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 17.63, df = 2, p-value = 0.0001485
```

```
> J6

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 4795.9, df = 2, p-value < 2.2e-16

> J7

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 37.602, df = 2, p-value = 6.838e-09

> J8

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 579.05, df = 2, p-value < 2.2e-16

> J9

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 6.4168, df = 2, p-value = 0.04042

> J10

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 87.298, df = 2, p-value < 2.2e-16
```

```
> J11

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 25.856, df = 2, p-value = 2.429e-06

> J12

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 1580.4, df = 2, p-value < 2.2e-16

> J13

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 16.556, df = 2, p-value = 0.000254

> J14

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 29.574, df = 2, p-value = 3.784e-07

> J15

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 1893, df = 2, p-value < 2.2e-16

> J16

        Jarque Bera Test

data:  coredata(assign(paste("viewts", col, sep = ""), zoo(dfcat[col],     as.Date(dates, format = "%y.%d.%m"))))
X-squared = 15861, df = 2, p-value < 2.2e-16
```

With all of this analysis, we created an ARMA model for the vehicles categories since it is the one that follows the behavior we saw in class in which p = 1 and q = 1.

```
> arima22 <- arima(coredata(dfd[1:100, 1]), order = c(1,0,2),include.mean = T, method = "ML")
> summary(arima22)

Call:
arima(x = coredata(dfd[1:100, 1]), order = c(1, 0, 2), include.mean = T, method = "ML")

Coefficients:
          ar1     ma1      ma2  intercept
      -0.7755  0.8887  -0.1113     0.3752
s.e.   0.0833  0.1236   0.1196     0.6101

sigma^2 estimated as 37.09:  log likelihood = -323.85,  aic = 657.7

Training set error measures:
                      ME     RMSE      MAE       MPE     MAPE     MASE         ACF1
Training set -0.005617947 6.089983 3.791988 -36.43638 310.4516 0.712884 -0.002764282
```

6. Evaluations and Results

6.1. Evaluation Methods

For all the classifiers we have built with the training data, we will compare the output of each of them with the labels specified at the training set from the data preprocessing step. We used a ten cross-fold

validation for trying to improve the initial performance of the models. Other way that we have to test the accuracy of the classifiers performance is the to us per-class precision, which would be more suitable for this kind of multi-class classification. This metric is particularly useful for this dataset because the class labels are not uniformly distributed (there are more videos that belongs to a certain category rather than to other).

We define the variable precision as the fraction of correct predictions for a certain class and recall as the fraction of instances of a class that where correctly predicted. We also calculate the F-1 score as the weighted average of precision and recall.

For the time-series model, we created a training and test model, in which we use an amount of days to use as training and the rest days to test the behavior of our model.

6.2. Results and Findings

If we compare the accuracy of predicting to which category a video belongs using the number of views and the likes, the results we obtain for the KNN and Naïve Bayes models are the following:

```
> ##### Naive Bayes
>
> #Cross-fold validation with NB
> nb2 =train(vid_norm_tr,train_lab,'nb',trControl=trainControl(method='cv',number=10))
> nb2_pred<-predict(nb2$finalModel,vid_norm_ts)
> #Accuracy
> 100*sum(nb2_pred$class == test_lab)/length(test_lab)
[1] 22.82609
> #Without cross-fold validation
> vid_train <- sapply(vid_train, as.numeric )
> vid_test <- sapply(vid_test, as.numeric )
> nb = naiveBayes(vid_norm_tr,train_lab,laplace=1)
> nb_pred = predict(nb,vid_norm_ts)
> #Accuracy
> 100*sum(nb_pred == test_lab)/length(test_lab)
[1] 12.64302

> ###### KNN modeling
> knn <- knn(train=vid_norm_tr, test=vid_norm_ts, cl=train_lab, k=27)
>
> #Accuracy
> 100*sum(knn == test_lab)/length(test_lab)
[1] 24.14188
>
> #Cross-fold validation with KNN
> knn2 =train(vid_norm_tr,train_lab,'knn',trControl=trainControl(method='cv',number=10))
> knn2_pred <- predict(knn2,vid_norm_ts)
> 100*sum(knn2_pred == test_lab)/length(test_lab)
[1] 22.54005
```

Decision tree results:

```
Overall Statistics

               Accuracy : 0.2534
                 95% CI : (0.2332, 0.2745)
    No Information Rate : 0.2466
    P-Value [Acc > NIR] : 0.2608

                  Kappa : 0.0743
 Mcnemar's Test P-Value : NA
```

The overall predictions are not positive since the best accuracy percentage we can reach is the one from the decision tree, around 25%. It is remarkable the improvement of the Naïve Bayes model performance when we use the cross-fold validation, but still the numbers are not high enough to assure a good prediction based on the dataset we are using.

Regarding the per-class validation, we have obtained the following results:

- For Naïve Bayes classifier:

```
#Per-class precision for Naive Bayes classifier
cm = as.matrix(table(test_lab,nb_pred))
n = sum(cm) # number of instances
nc = nrow(cm) # number of classes
diag = diag(cm) # number of correctly classified instances per class
rowsums = apply(cm, 1, sum) # number of instances per class
colsums = apply(cm, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes
precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)
data.frame(precision, recall, f1)
```

```
    precision     recall         f1
1         NaN 0.00000000        NaN
2         NaN 0.00000000        NaN
10 0.58620690 0.07488987 0.13281250
15 0.00000000 0.00000000        NaN
17        NaN 0.00000000        NaN
19        NaN 0.00000000        NaN
20        NaN 0.00000000        NaN
22        NaN 0.00000000        NaN
23 0.12820513 0.03225806 0.05154639
24 0.28205128 0.07656613 0.12043796
25 0.10530421 0.96428571 0.18987342
26 0.13181818 0.17791411 0.15143603
27 0.02325581 0.01666667 0.01941748
28 0.05882353 0.00990099 0.01694915
29        NaN 0.00000000        NaN
43        NaN 0.00000000        NaN
```

As it was expected, the model predicts better those videos that belong to the Music category (which is the category with more number of videos in the dataset) with a 58,62% of true positives.

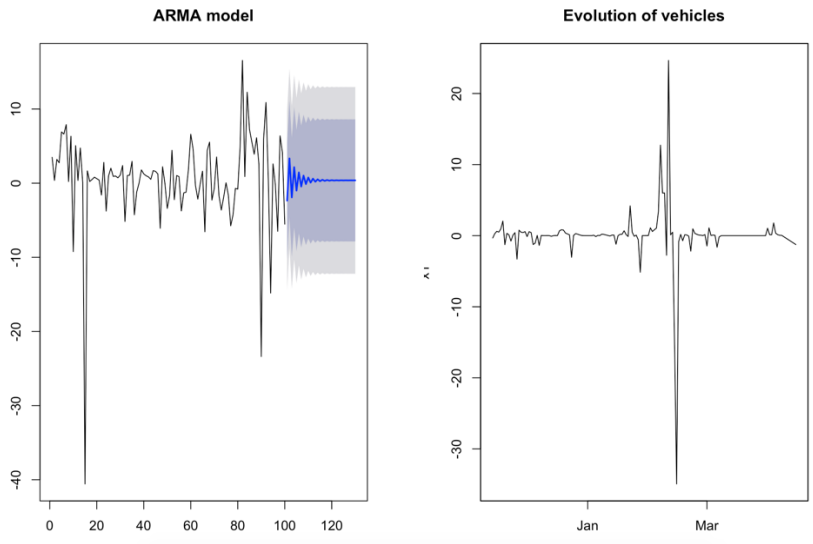
- For the K-Nearest Neighbors:

```
#Per-class precision for KNN classifier
cm = as.matrix(table(test_lab,knn2))
n = sum(cm) # number of instances
nc = nrow(cm) # number of classes
diag = diag(cm) # number of correctly classified instances per class
rowsums = apply(cm, 1, sum) # number of instances per class
colsums = apply(cm, 2, sum) # number of predictions per class
p = rowsums / n # distribution of instances over the actual classes
q = colsums / n # distribution of instances over the predicted classes
precision = diag / colsums
recall = diag / rowsums
f1 = 2 * precision * recall / (precision + recall)
data.frame(precision, recall, f1)
```

	precision	recall	f1
1	0.00000000	0.000000000	NaN
2	NaN	0.000000000	NaN
10	0.34285714	0.264317181	0.29850746
15	NaN	0.000000000	NaN
17	0.11111111	0.007142857	0.01342282
19	NaN	0.000000000	NaN
20	NaN	0.000000000	NaN
22	0.14285714	0.020547945	0.03592814
23	0.08163265	0.025806452	0.03921569
24	0.26969416	0.675174014	0.38543046
25	0.18333333	0.235714286	0.20625000
26	0.15853659	0.159509202	0.15902141
27	0.05357143	0.050000000	0.05172414
28	0.11111111	0.009900990	0.01818182
29	NaN	0.000000000	NaN
43	NaN	0.000000000	NaN

Same as before, the model predicts better those videos that belong to the Music category, but in this case the true positives rate is significantly lower, only a 34,28%.

For the time-series analysis, as we commented in the analysis part, we created an ARMA model with p = 1 and q = 1. To test the behavior, we used the first one hundred days of our vehicles class to predict what happens afterwards.



We can observe that even though the model is not precise in the actual values we can observe that it predicts the overall trend that this category is following.

7. Conclusions and Future Work

7.1. Conclusions

The main conclusion we have reached in this project Is that real life problems are difficult to predict and study, and the dataset we have been working on is huge and spread. One possible solution for a better outcome would be to tray to narrow down the data more so the predictive modeling will be more successful.

As we said in our Project Proposal, we could not state a clear relation between the variables we selected for an accurate guessing for classification. The results obtained for the classifiers are not good enough to say with conviction that we can predict to which category a video belongs to from the information we have available.

For our time-series analysis we can conclude that it is difficult to predict the exact numbers that a category will have the following days. Sometimes, what we want to know is not the exact value but the trend or pattern that the object under study follows. As an example, we could put the stock market in which, even though the amount of variation of a stock can be relevant, what matters is the trend will follow the company to know if someone wants to invest or sell in stocks for a determined company.

Finally, we conclude that trending videos and categories are really difficult to predict with only numerical data, but maybe the results would improve by using other qualitative data that we have available at the dataset. Altogether, this project has been an interesting learning experience and a good way to put in practice all we have learned on the course.

7.2. Limitations

The most important limitations we have faced along the completion of this project were that the dataset was not standardized and that the numerical features were scarce. It is a huge dataset and we did a lot of preprocessing before diving in the modeling section. Moreover, the numerical features that we had

worked are not enough for making good predictions, and we believe that maybe a predictor that combines numerical features and some kind of dictionary classifier would have given us better results.

7.3. Potential Improvements or Future Work

The principal improvements we see for future lines of this project would be the use of dictionary classification. The dataset has many textual variables that can be used in order to predict if a certain video belongs to a category because the title, description and tags of a video contain key words for a certain category. We can also use these features for a sentimental analysis, where we could say if a video is happy or aggressive or even if the probability to be trending is higher because the title contains a word that most of the trending videos of that category have it too.