data analysis

sheena

2019/3/27

With the expansion of the Internet, more and more people enjoys reading and sharing online news articles. The number of shares under a news article indicates how popular the news is. In this project, we intend to find the best model and set of feature to predict the popularity of online news, using machine learning techniques. We implemented 6 different learning algorithms on the dataset, ranging from K Nearset Neighbour to Random Forest. Their performances are recorded and compared. Feature selection methods are used to improve performance and reduce features. Random Forest turns out to be the best model for prediction, and it can achieve an accuracy of 65% with optimal parameters. Our work can help online news companies to predict news popularity before publication.

```
library(tidyverse)
library(readr)
library(ggplot2)
library(ggrepel)
library(PredPsych)
library(e1071)
library(corrplot)
library(lattice)
library(caret)
library(MASS)
library(plyr)
library(class)
library(randomForest)
library(factoextra)
```

1.Data Exploration

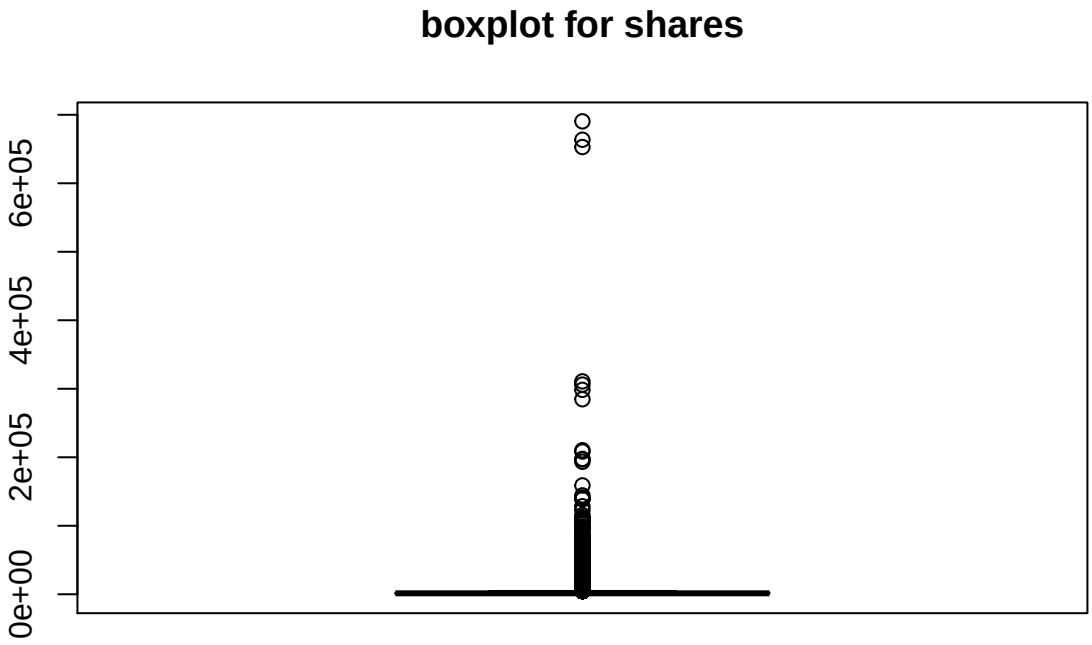
```
df<-read.csv('OnlineNewsPopularity_1.csv')
```

Our target variable is based on feature “shares” so I give the summary in the first step

```
summary(df$shares)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     930    1400    2929    2500  690400
```

```
boxplot(df$shares,main="boxplot for shares", xlab="shares")
```



The boxplot intends to show distribution of our target variable. It is obvious there are plenty of "outliers" which we may not consider.

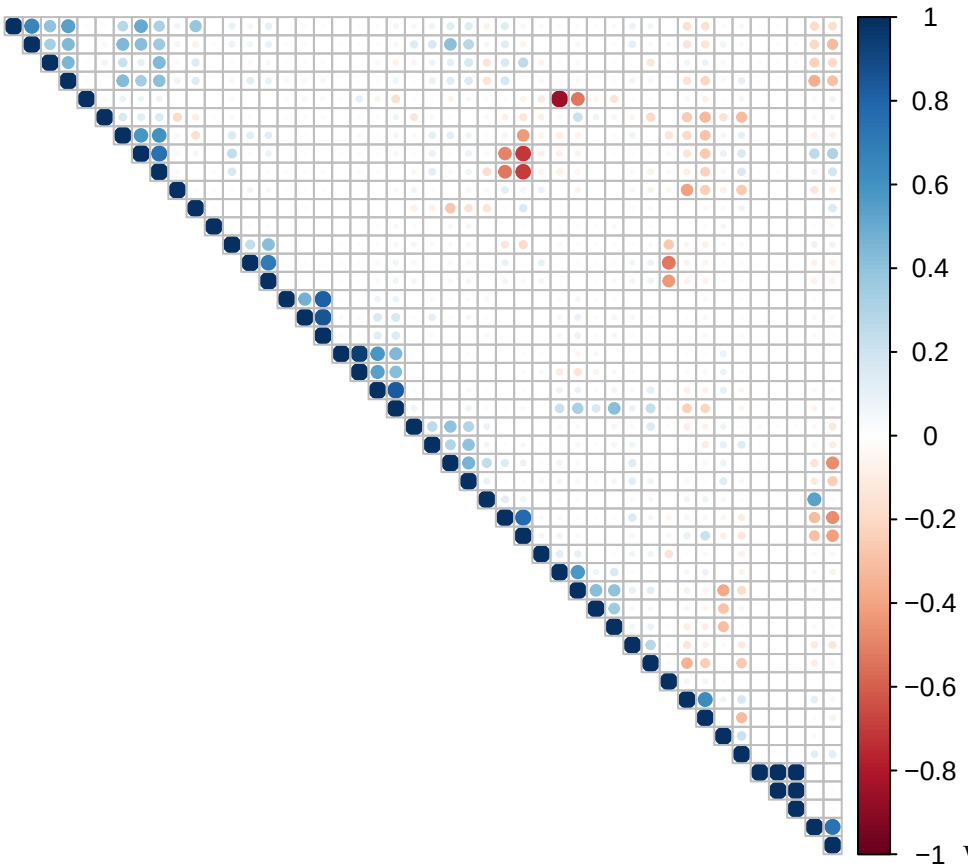
```
df2 <- df[-which(df$shares>10000),]
df2$class <- as.factor(ifelse(df2$shares<=1400, "0", "1"))
#head(df2)
```

Target variable "class" has two values 0 and 1, which is based on "shares". For news with share number smaller than 1400, we regarded it as "not popular" and assign 0 to it; otherwise, we regarded it as "popular" and assigned 1 to it.

The above coding is the visualization of data which enables us to assign appropriate value to each news.

2. Feature Selection

```
corr<-cor(df[,1:46])
corrplot(corr,  tl.pos='n',type="upper", order="hclust")
```



We find the size of feature is large and some features are highly correlated.

Fisher Score is a good way to choose feature. A larger Fisher Score indicates greater discriminative power of certain feature. Here I use the original dataset so we can cover all information.

```
f1 <- as.data.frame(fscore(df,48,1:46))
```

```
##
## Performing Feature selection f-score analysis
```

```
f <- as.data.frame(cbind(colnames(df[1:46]),f1))
f <- f[order(f[,2],decreasing=T),]
colnames(f) <- c("feature","fisher score")
f[1:25,]
```

```
##                                           feature fisher score
## LDA_04                                     LDA_04       0.0670
## day                                           day       0.0500
## LDA_02                                     LDA_02       0.0480
## kw_avg_avg                              kw_avg_avg       0.0320
## LDA_01                                     LDA_01       0.0240
## LDA_00                                     LDA_00       0.0230
## global_sentiment_polarity   global_sentiment_polarity       0.0230
## rate_negative_words               rate_negative_words       0.0210
## rate_positive_words               rate_positive_words       0.0180
## kw_min_avg                              kw_min_avg       0.0160
## global_rate_positive_words global_rate_positive_words       0.0160
## num_hrefs                                num_hrefs       0.0150
## kw_min_min                              kw_min_min       0.0100
```

```
## global_subjectivity               global_subjectivity       0.0100
## n_unique_tokens                       n_unique_tokens       0.0092
## num_self_hrefs                         num_self_hrefs       0.0088
## n_tokens_content                     n_tokens_content       0.0083
## num_keywords                             num_keywords       0.0078
## self_reference_max_shares   self_reference_max_shares       0.0078
## self_reference_avg_sharess self_reference_avg_sharess       0.0078
## n_non_stop_unique_tokens     n_non_stop_unique_tokens       0.0071
## n_tokens_title                         n_tokens_title       0.0068
## title_sentiment_polarity     title_sentiment_polarity       0.0062
## global_rate_negative_words global_rate_negative_words       0.0057
## min_positive_polarity           min_positive_polarity       0.0055
```

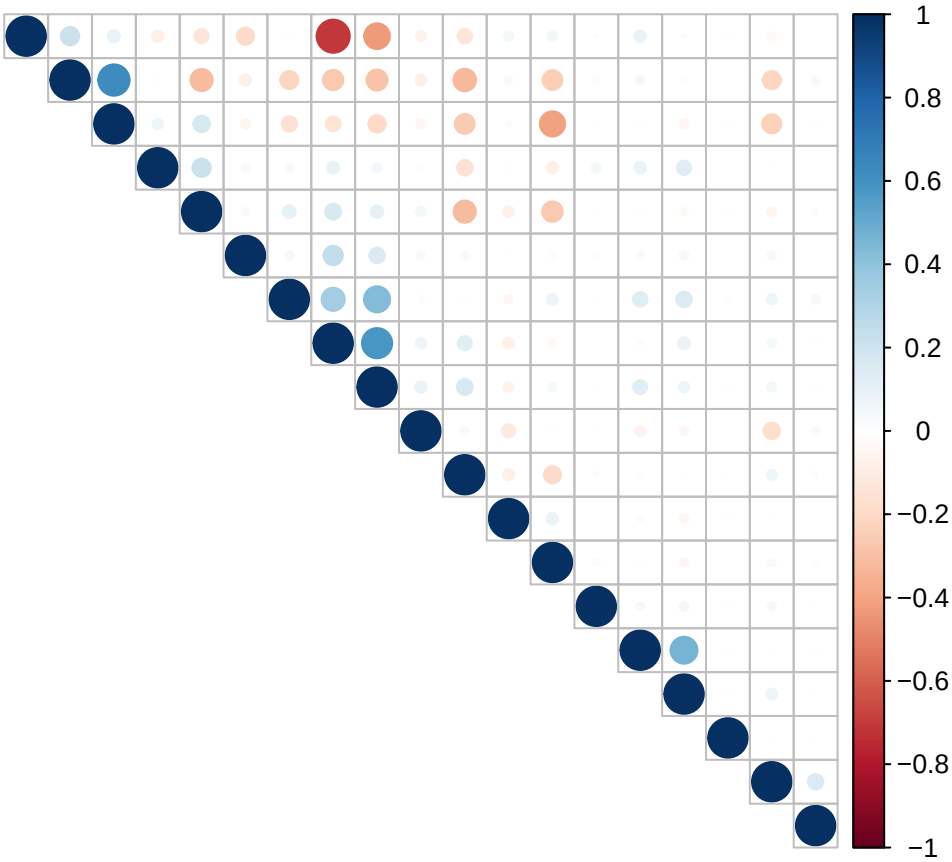
Then I remove some high-correlated features and just keep 19 features, save as "OnlineNewsPopularity-Cleaned.csv".

```
df_final<-read_csv('OnlineNewsPopularityCleaned.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
corr1<-cor(df_final[,1:19])
corrplot(corr1,  tl.pos='n',type="upper", order="hclust")
```



3.Classification

In this step I perform 6 statistic algorithms: K Nearst Neighbour, Navies Bayes, Quadratic Discriminant

Analysis, Linear Discriminant Analysis, Logistic Regression, Random Forest to calssify dataset.

3.1 split dataset

I split dataset into test and train sets to examine their performance by calculating the cross validation score.Train set contains 70% of the observation while test set contains the remaining 30%.

```
df_final$class <- as.factor(ifelse(df_final$shares<=1400, "0", "1"))
```

```
CVgroup <- function(k,datasize,seed){
  cvlist <- list()
  set.seed(seed)
  n <- rep(1:k,ceiling(datasize/k))[1:datasize]    #split dataset into k fold
  temp <- sample(n,datasize)   #???
  x <- 1:k
  dataseq <- 1:datasize
  cvlist <- lapply(x,function(x) dataseq[temp==x])  #dataseq
  return(cvlist)
}
k <- 10
datasize <- nrow(df_final)
cvlist <- CVgroup(k = k,datasize = datasize,seed = 1206)
#cvlist
```

3.2 Classification

Then I perform classification algorithms on the train dataset and predict their performance on test dataset. The results are stored in “s”

```
score<-function(x_train,x_test,y_train,y_test){
  time0 <- Sys.time()
  #y_train <- y_train[,1,drop=TRUE]
  #y_test <- y_test[,1,drop=TRUE]
  #KNN
  cl<-y_train
  pred_knn<-knn(train=x_train,test=x_test,cl,k=10)
  s1<-mean(pred_knn==y_test)

  #Naive Bayes
  nb<-naiveBayes(x=x_train,y=y_train)
  pred_nb<-predict(nb,x_test)
  s2<-mean(mean(pred_nb==y_test))

  #QDA
  qda<-qda(y_train~.,data=x_train)
  pred_qda<-predict(qda,newdata= x_test)$class
  s3<-mean(mean(pred_qda==y_test))

  #LDA
  lda<-lda(y_train~.,data=x_train)
  pred_lda<-predict(lda,newdata= x_test)$class
  s4<-mean(mean(pred_lda==y_test))
```

```
  #Logistic Regression
  lg <-  glm(y_train~., data=x_train, family=binomial)
  lgc <-  update(lg, ~.)
  pred_lg <- predict(lgc, newdata= x_test, type="response")
  pred_lg <- ifelse(pred_lg > 0.5,1,0)
  s5 <- mean(pred_lg == y_test)
  #Random Forest
  rf <- randomForest(y_train~.,data=x_train,ntree=300,mytry=15)
  pred_rf <- predict(rf,newdata=x_test,type="class")
  s6 <- mean(pred_rf==y_test)
  time <- Sys.time()-time0

  return(cbind(s1,s2,s3,s4,s5,s6,time))
}
for (i in 1:10){
  train <- df_final[-cvlist[[i]],]
  test <- df_final[cvlist[[i]],]
  feature_train<-train[,1:19]
  target_train<-train[,21][,1,drop=TRUE]
  feature_test<-test[,1:19]
  target_test<-test[,21][,1,drop=TRUE]
  s <- score(feature_train,feature_test,target_train,target_test)
}
```

```
s <- as.data.frame(s)
s
```

```
##          s1        s2       s3        s4        s5        s6     time
## 1 0.5816174 0.5341689 0.529991 0.6284691 0.6320501 0.6532378 46.99321
```

It shows Random Forest achieves the highest accuracy among six algorithms.

3.3 Feature Importance

```
train <- df_final[-cvlist[[1]],]
test <- df_final[cvlist[[1]],]
feature_train<-train[,1:19]
target_train<-train[,21][,1,drop=TRUE]
  #feature_test<-test[,1:19]
  #target_test<-test[,21][,1,drop=TRUE]
rf <- randomForest(target_train~.,data=feature_train,
                     ntree=300,mytry=15,keep.forest=FALSE, importance=TRUE)
df_rf <- as.data.frame(importance(rf))
```

The feature importance is then calculated from the Random Forest model. And it shows that the "day" is the most important feature and "kw_avg_avg" is the second most important feature.

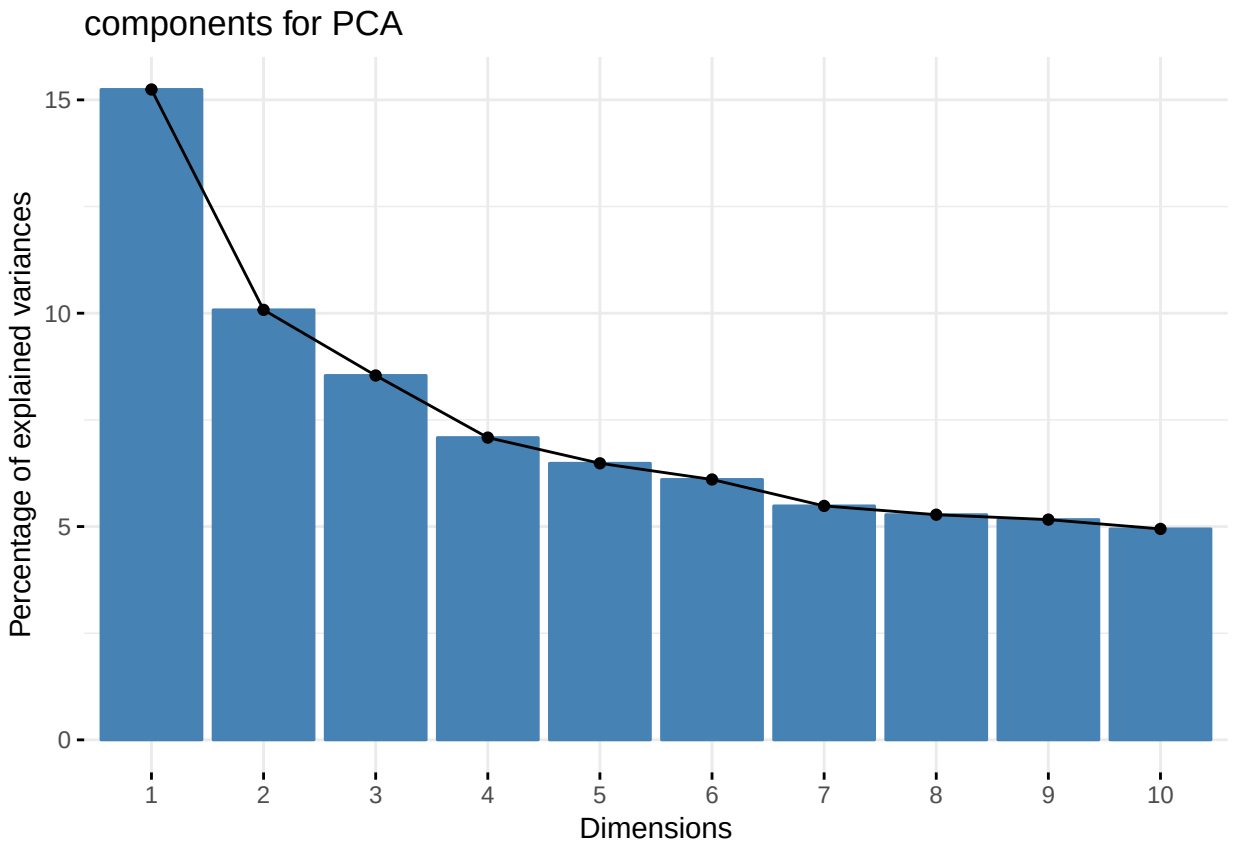
4. PCA

In order to improve the accuracy, I try PCA and calculate the best number for components. As suggested by the histgram plot "components for PCA", the transformed data with PCA just need keep 3 components since

the ratio does not increase when the number of components goes beyond 3.

```
pca <- prcomp(df_final[,1:19], scale =TRUE)

fviz_eig(pca)+labs(title = "components for PCA")
```



Then perform classification on dimension-reduced dataset

```
df_pca <- as.data.frame(cbind(pca$x[,1:3],df_final[,21]))
for (i in 1:10){
  train_pca <- df_pca[-cvlist[[i]],]
  test_pca <- df_pca[cvlist[[i]],]
  feature_train_pca<-train_pca[,1:3]
  target_train_pca<-train_pca[,4]
  feature_test_pca<-test_pca[,1:3]
  target_test_pca<-test_pca[,4]
  s_pca <- score(feature_train_pca,feature_test_pca,target_train_pca,target_test_pca)
}
```

```
df_s <- rbind(s_pca,s)
colnames(df_s) <- c("Knn","Naive Bayes","QDA","LDA","Logistic Regression","Random Forest","time")
rownames(df_s) <- c("with PCA","without PCA")
df_s
```

```
##                  Knn Naive Bayes       QDA       LDA Logistic Regression
## with PCA    0.5675918   0.5663981 0.5687854 0.5684870           0.5693823
## without PCA 0.5816174   0.5341689 0.5299910 0.6284691           0.6320501
##             Random Forest     time
## with PCA        0.5717696 16.63925
## without PCA     0.6532378 46.99321
```

From the results we can find “time” decreases a lot. PCA is a good way to reduce dimension, which could give us a lower dimensional approximation for original dataset while preserving as much variability as possible.

However, cross validation score drops obviously after applying PCA, especially when it comes to “LDA”,“Logistic Regression”,“Random Forest”, which means PCA is not a good choice to improve accuracy. I think the reason PCA performes badly is that original feature set is well-designed and correlated information between features is limited. Our feature selection is very successful so PCA just results in losing information.

5.Future Work

Conclusively , this project was focused on predicting the popularity of news articles based on the number of shares of the articles. Given the high dimensionality of our data, we used fisher score to select the most relevant features. We then used several classification algorithms to classify the data. Our results show that the accuracy of the classification models is quite low with Random Forest being the highest with an accuracy of 65%. We then used PCA to process the data but it failed to improve the classification as expected. To improve accuracy, there is limited room in model selection, but much room in feature selection. Although the original dataset contains 61 features, many of them are highly correlated and thus are removed. We believe that by adding additional features that could describe the news effectively, we will be able to improve the accuracy. In the future, we could implement an efficient data cleaning method instead of manually removing and deleting features with missing data during the preprocessing stage. Also, due to the size of the data and limited computing power, we could not run some classification models like SVM. In the future we could use a computer with higher computation power to run these other classification models and see how they compare to our current results.