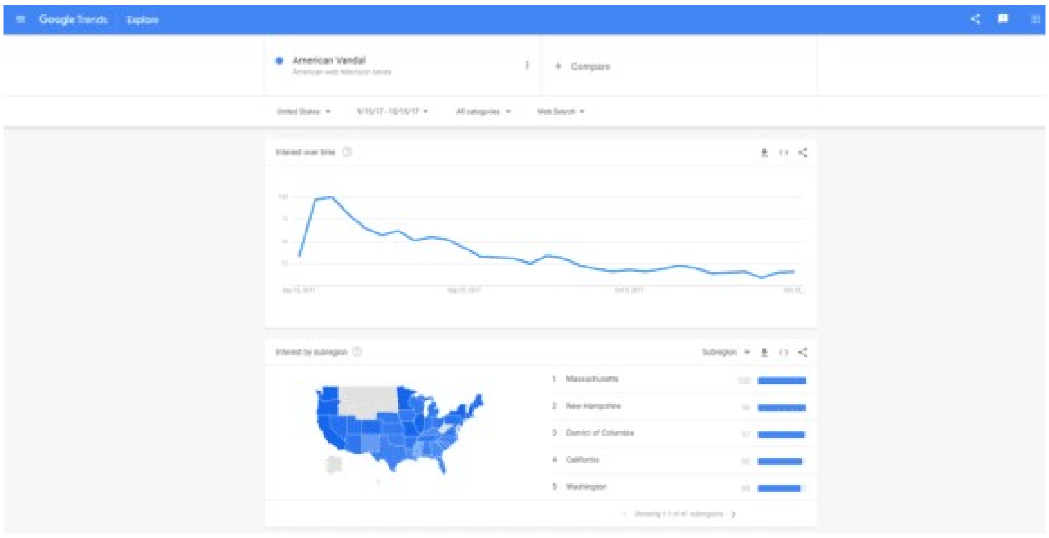
Step 1: Data identification

Firstly, I needed to make a compilation of all the shows Netflix has renewed or cancelled in the past. I was able to get a brief list through a google search and stumbled up Business Insider. They had compiled a list of shows that were cancelled. I eliminated all Marvel/Disney shows in my data collection because “Disney+” please!.

Step 2: Data collection

Trends

One of the main factors that contribute for Netflix renewing the show is the view count. Netflix does not publish these numbers publicly with few exceptions. I identified all the shows, their season and release dates. With this compiled list, I was able to grab google trend data for each of this show. I looked at google trend for 30 days since the release of the show as one of my primary variable. Below is an example on how you could get the data from google trends.



Reviews

Netflix is one of those companies that make shows or movies irrespective of what critics think. I am a little biased here. I usually believe in user reviews over critics. Just for this study, I did include both user and critic scores from rotten tomatoes

Release

There are different aspects to when a show is release. For example if it’s release on a Friday or Tuesday. Also, we need to consider what month or season the show was released. That’s exactly what I did. I captured the day show was released, month and season of the year as my third variable.

Time between releases

Final variable that I considered was, the time between each subsequent season releases. This variable was used mostly as a test hypothesis to see if this affected in any way.

Here is the link to the file if you want to download the data.

netflix-dataDownload

Step 3: Data Analysis

Missing Values

While I was collecting data, I made a mistake of not collecting trend data for 21 days for a particular title. I rather collected it for 19 days. Rather than going back and collecting it, I imputed the vales to mean. I also removed the show title name as it adds no value to the analysis.

I have also split the data into modeling data and final. Data contains information for training model and final is our data we will use in the end to predict if the show "Indian Matchmaking" will be renewed or not.

```
library(caret)
library(dplyr)
library(ModelMetrics)

# load data
data = read.csv("C:\\Users\\aanamruthn.NADENSO\\Downloads\\nf.csv")

data = data %>% mutate_all(~ifelse(is.na(.x), mean(.x, na.rm = TRUE), .x))
data = data %>% select(-c("Show", "release_date"))

# convert variable to factor
data$renewed = as.factor(data$renewed)
# 2 is "yes" 1 is "no"

# split model building and prediction
final = data[42, ]
data = data[1:41,]
```

Step 4: Modeling Simulation

Model simulation is a process that I have traditionally followed for a while now. In this process, we will build a series of 30 models with random seed values and train-test splits. We will also tune the models through a grid search. The main reason for doing this is that, we want to make sure we get near consistent results rather than a 1 time luck. As for the model, we will use linear SVM.

```
# function to train models
build_models = function(i){
    # set random seed values
    set.seed(sample(1:1000, 1))

    # create data partition
    split = runif(1, 0.6, 0.9)
    samp = createDataPartition(y = data$renewed, p = split)
    x = data[samp$Resample1, ]
    y = data[-samp$Resample1, ]

    #10 folds repeat 3 times
    control = trainControl(method='repeatedcv',
                           number=10,
                           repeats=5)

    #Metric compare model is Accuracy
    metric = "Accuracy"
    set.seed(sample(1:1000, 1))

    # SVM cost grid
    svmgrid = expand.grid(cost=seq(0.05,1,by = 0.05))

    # train the model
    model = train(renewed~.,
                        data=x,
                        method='svmLinear2',
                        metric='Accuracy',
                        tuneGrid=svmgrid,
                        trControl=control,
                        importance = TRUE,
                        preProcess= c("center", "scale")
                       )
    # print the results of the model
    print(model)

    # get the metrics for test set
    test = caret::confusionMatrix(factor(y$renewed), predict(model, y))

    # return the simulation results
    return(data.frame(train_accuracy = max(model$results$Accuracy),
                      test_accuracy = as.numeric(test$overall[1]),
                      AUC = auc(factor(y$renewed), predict(model, y)),
                      train_split = split))
}

# build the model 30 times
sim_Results = do.call(rbind, lapply(1:30, build_models))
```

The simulation results are as shown below.

```
> sim_Results
   train_accuracy test_accuracy       AUC train_split
1       0.7966667     0.6923077 0.6375000   0.6661155
2       0.7900000     0.7333333 0.7500000   0.6029180
3       0.8366667     0.7000000 0.7083333   0.7334306
4       0.7716667     0.6000000 0.5416667   0.7295723
5       0.8683333     0.6428571 0.6777778   0.6295557
6       0.7850000     0.8571429 0.8888889   0.6321844
7       0.7366667     0.8000000 0.7916667   0.7398796
8       0.8633333     0.6363636 0.6071429   0.6932221
9       0.8416667     0.7500000 0.8000000   0.7601317
10      0.8633333     0.6666667 0.6944444   0.6184694
11      0.8500000     0.5384615 0.5125000   0.6796205
12      0.7950000     0.8000000 0.7500000   0.8747213
13      0.7433333     0.7500000 0.8000000   0.7768006
14      0.8093333     0.6000000 0.5833333   0.8490845
15      0.7506667     0.8000000 0.8333333   0.8529004
16      0.7000000     0.6000000 0.5555556   0.6168192
17      0.7936667     0.8333333 0.8750000   0.8221747
18      0.7333333     0.7692308 0.7375000   0.6597461
19      0.8150000     0.5833333 0.5571429   0.6855324
20      0.7966667     0.8000000 0.8333333   0.7477105
21      0.7473333     1.0000000 1.0000000   0.8951201
22      0.6933333     0.8000000 0.7500000   0.6090903
23      0.7683333     0.7500000 0.7285714   0.6824318
24      0.7333333     0.9000000 0.9166667   0.7218617
25      0.7066667     0.5384615 0.5125000   0.6619311
26      0.7380000     0.8000000 0.8333333   0.8455452
27      0.8180000     0.6000000 0.6666667   0.8500800
28      0.7783333     0.8000000 0.7916667   0.7385316
29      0.7863333     0.8333333 0.8750000   0.8320224
30      0.7733333     0.7777778 0.8333333   0.7509293
```

From the above simulation results we can notice that the results are pretty consistent with train and test sets.

It might be quite a lot to view all the model building results at once. So, lets do a summary of the results for easier understanding as shown below. Here we see that the avg train accuracy is around 78% and test accuracy is 73% which is not too bad for a model with smaller sample size. The AUC is around 73% as well.

```
sim_Results %>% summarize(avg_train_accuracy = mean(train_accuracy),
                            avg_test_accuracy = mean(test_accuracy),
                            avg_AUC = mean(AUC)
                            )
  avg_train_accuracy avg_test_accuracy   avg_AUC
1          0.7827778         0.7317534 0.7347619
```

Step 5: Variable Importance

We currently have an acceptable model for predicting if the show will be renewed or not. Just out of curiosity, we also need to know the most significant factors for this predicting this/or the most important variable contribution for prediction. We will use caret's feature to do this. The function automatically scales the importance scores to be between 0 and 100. Since we used SVM classification model, the default behavior is to compute the area under the ROC curve to get variable importance.

From the below results we see the top 20 important features. Season was the most important factor followed by the days between season, Day.6 trend on google searcher and critic_score.

```
# variable importance for the model
varImp(model)
ROC curve variable importance

  only 20 most important variables shown (out of 38)

                     Importance
Season                   100.00
days_between_seasons      75.61
Day.6                     72.76
critic_score              53.66
Day.7                     50.00
Day.5                     46.34
Day.24                    45.12
Month                     40.65
Season.1                  30.89
Day.8                     28.46
Day.13                    27.24
Day.12                    25.61
Day.2                     24.80
Day.18                    23.58
Release_day               22.36
Day.23                    17.89
Day.21                    17.48
Day.15                    16.26
Day.20                    14.63
Day.0                     14.63
```

Step 6: What we have all been waiting for!

Finally, we will use our show data to predict if "Indian Matchmaking" show would be renewed or not. From the prediction results, we can see that the show is most likely to be renewed for season 2.

```
# predict the results
predict(model, final)
# [1] 2
# 2 is "yes" 1 is "no"
```

Reflection Points

If I would go back and work on this model, there are three things I would definitely do differently.

1. Data collection: I would add more variables like genre, budget, country, twitter trends and probably increase my sample size from few shows to at least 30.
2. Feature extraction: performing a polynomial expansion on the data would be a great expansion of features.
3. Modeling: I would train of different types of models such as trees, boosting, neural networks etc and compare the results.

Conclusion

I honestly don’t know what procedure Netflix uses to renew their shows. From the above analysis we can notice that we are not that off. If we could tie into Netflix’s viewership and monthly trend data we could definitely build a more accurate model that predicts if Neflix should renew the show or not.