MS&E 226 Mini-Project 3
Joseph Wu & Min Cheol Kim

Prediction on the test set

Regression prediction on test set

Best Model: OLS	Prediction Error (RMSE)
Train Set	157.44
Test Set	190.62

We used RMSE (Root Mean Squared Error) to measure the prediction error of the regression model on both train and test sets. We used cross validation method to estimate the true error from the train set. Notice that the prediction error is still quite a bit higher than the prediction error on the train set, even with the cross validation method. However, the prediction error is lower than our baseline error, showing that our regression model indeed improves prediction accuracy.

Classification prediction on test set

Best Model: Logistic Regression	Accuracy
Approximation of Test Accuracy	0.79
Calculated Test Accuracy	0.80

The true test error when our fitted logistic regression model was applied to the brand-new test set was very close to our approximation of test accuracy. This gives us confidence that what we say about our model just using the training data generalizes to the world outside of the test data relatively well.

This test accuracy may not hold if a significant amount of time passes between the time when the training data set was collected and the test data set was collected. Here, we used datasets that are a week apart, but if the time difference grows to weeks and months, we may need to refit the model to make the training data better look like the incoming, new data. We discuss this further in the discussion.

Inference

We decided to implement the following inference methods on our regression task (predicting the continuous variable - number of peak viewers during a streaming session). See our linear regression model below:

	Coefficient	P-Value
(Intercept)	1.486052e+00	0.40193

Video_bitrate	1.564666e-03	0.00115
Uptime	-1.251898e+00	0.53251
Uptime_sec	2.203679e-04	4.43e-15
Channel_view_count	5.915931e-05	< 2e-16
Featured	4.388593e+01	2.32e-14
Channel_subscription	6.846699e+02	< 2e-16
Mature	-3.804961e-01	0.81848
US	-2.598272e+00	0.03061
Eng_lang	2.256840e+00	0.15083

Part A) Using a p-value threshold of 0.05, we find that Video_bitrate, Uptime_sec, Channel_view_count, Featured, Channel_subscription, and the label US to be statistically significant coefficients. Many of these coefficients, especially channel_view_count, featured, and channel_subscription, are readily interpretable. Here, significance for a coefficient means that if this coefficient was actually 0 (the covariate does not affect viewer_count in any way) there is a p-value chance of seeing the data as it happened. We can believe the results for the covariates mentioned above, but video_bitrate, uptime_sec, and the US label seem to be rather spurious, and are not as readily interpretable.

Part B) After fitting our model on the test data, we looked at the coefficients and their corresponding P values. Although the coefficients are slightly different, the same significant coefficients held (uptime, featured, channel_subscription, mature, US, eng_lang), showing that indeed these attributes have a strong correlation to the number of viewers who tune in during a session.

Part C) After bootstrapping 5000 times, we used the standard error from bootstrap distribution to estimate a 95% Confidence Interval for each coefficient:

	95% Confidence Interval	
(Intercept)	1.486052e+00	± 4.199989e+00
Video_bitrate	1.564666e-03	± 7.430702e-04
Uptime	-1.251898e+00	± 3.133581e+00
Uptime_sec	2.203679e-04	± 7.947176e-05
Channel_view_count	5.915931e-05	± 2.86392e-05
Featured	4.388593e+01	± 7.704256e+01
Channel_subscription	6.846699e+02	± 1.26249e+02
Mature	-3.804961e-01	± 3.745552e+00
US	-2.598272e+00	± 2.334811e+00
Eng_lang	2.256840e+00	± 4.119952e+00

Side by side comparison of R generated Std Error and Bootstrap Std Error:

	R Std Error	Bootstrap Std Error
(Intercept)	1.773e+00	2.142851e+00
Video_bitrate	4.813e-04	3.791174e-04
Uptime	2.006e+00	1.598766e+00

Uptime_sec	2.810e-05	4.054682e-05
Channel_view_count	5.408e-07	1.461184e-05
Featured	5.750e+00	3.930743e+01
Channel_subscription	8.110e+00	6.441278e+01
Mature	1.658e+00	1.910996e+00
US	1.202e+00	1.19123e+00
Eng_lang	1.571e+00	2.102016e+00

We notice slight differences in the standard error of every coefficient but the R generated std error and Bootstrap std error are generally in the same neighborhood. The difference result because R employs a different methodology to estimate the std error than the Boostrap method. R relies on the single sample input into the linear model to estimate the std error of the coefficients generated. Bootstrap is a way to artificially generate lots of samples from the original sample and uses the standard deviation of the distribution to estimate the std error.

Part D) Because we have either removed or converted irrelevant variables and categorical variables, we reduced the number of covariates to 9. Adding back all the covariates that can be input into OLS is essentially creating noise in the algorithm. The new coefficients with all the covariates fitted show minimal changes in the original coefficients:

	Coefficient (9 Cov)	Coefficient (14 Cov)
(Intercept)	1.486052e+00	1.424352e+00
Video_bitrate	1.564666e-03	1.335672e-03
Uptime	-1.251898e+00	-1.244891e+00
Uptime_sec	2.203679e-04	2.203679e-04
Channel_view_count	5.915931e-05	5.912264e-05
Featured	4.388593e+01	4.438519e+01
Channel_subscription	6.846699e+02	6.832492e+02
Mature	-3.804961e-01	-3.603943e-01
US	-2.598272e+00	-2.342986e+00
Eng_lang	2.256840e+00	2.003835e+00
Session_id		2.536339e-02
Channel_id		8.795637e-04
Video_width		5.758951e-04
Embed_count		5.221779e-05
Site_count		2.2359678e-04

Part E) We found multiple coefficients as significant, and given that we started out with 13+ coefficients, it would not be unlikely that at least one of the coefficients we found significant is actually a false positive. The Bonferroni correction would make our effective pvalue around 0.006, in which case the video_bitrate and the US label would not make it as significant coefficients. This actually seems reasonable, because we were not completely convinced that these variables display true significance. Our model building involved a subjective step where we incorporated variables that we thought were significant, but this step did not use our data; in other words, our model selection procedure did not involve going back to the data multiple times.

Because of this, we are fairly confident that we are not subject to post selection inference, aside from the possibility that some actually significant covariates were excluded based on our false intuition

Part F) In our context, the “featured” variable indicated whether Twitch TV had shown the session on the first landing page of this website (Figure 1). From this exogenous data and our significant coefficient and p value, we are inclined to say that there is a causal relationship between being featured and the viewer count of a session. Being on the prime real estate of the website’s landing page would reasonably lead to more publicity and promotion, thus more viewers. There may be other variables in the data that lead to a session being featured. However, we have no idea what the selection criteria for a session to be featured on Twitch TV’s home page; therefore the featured variable is not representative of a random assignment. Because we may have a sampling bias that we cannot confirm or deny, just using our dataset, we cannot give quantitative evidence for causality even between the featured variable and viewer count.

Additionally, we are inclined to assert a causal relationship between whether a channel takes subscriptions (the “channel_subscription” variable) and the viewer count of its session. On Twitch, streamers have the option to make their channel into a subscription channel, where the audience can subscribe and make donations to. Subscribers of a channel can view the streams when logging into their account and receive notifications of new streams (Figure 2). This gives a huge advantage in terms of reaching fans and promoting a channel’s content. Since this subscription option is an independent decision that a streamer makes when creating a channel, we don’t believe there are confounding variables within the data. An argument can be possibly made that streamers who take subscribers produce more consistent and better content. However, these are variables that are outside of the data we used.

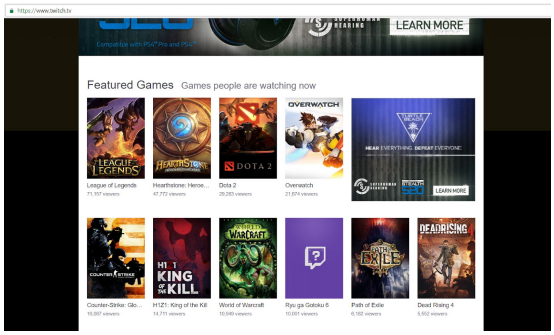


Figure 1

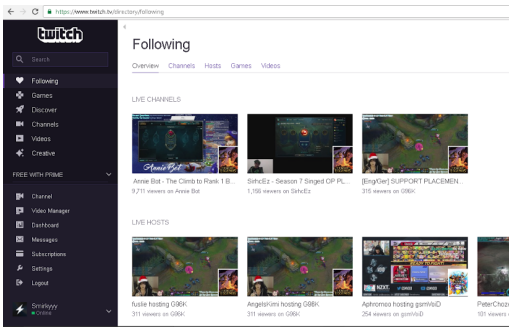


Figure 2

Discussion

Our dataset comes from an online streaming TV source. As the administrators of the website, there could be many questions that they would like to answer about the user behaviors on the platform. These questions would be answered by both prediction and inference. For example, they may simply want to predict the viewer count in various sessions held at specific times, to estimate what kind of backend server bandwidth they need; depending on how their backend is

implemented, it could lead to business decisions involving hardware rental/purchase. Our regression model for prediction could be a starting step in answering this question. In addition, the site admins may want to find out what different factors make a significant role in contributing to the viewer count, so they can focus on that factor if they are able to. One example might be adjusting their criteria for promoting “hot” or “featured” sessions, so that the ones that actually do get featured can get maximum benefit in terms of viewer count.

Our model would hold up within intervals of time with similar properties. For example, a model fitted for the holiday season would probably hold up for the next few months when people maintain their same habits before the holidays are over; the model would most likely have to be refitted every time there is a significant seasonal change that prompts people to change their behaviors. In addition to seasons, when there are important gaming events (League of Legends tournaments, etc.) our model may have to be refitted to reflect that people who usually do not use the platform are now active.

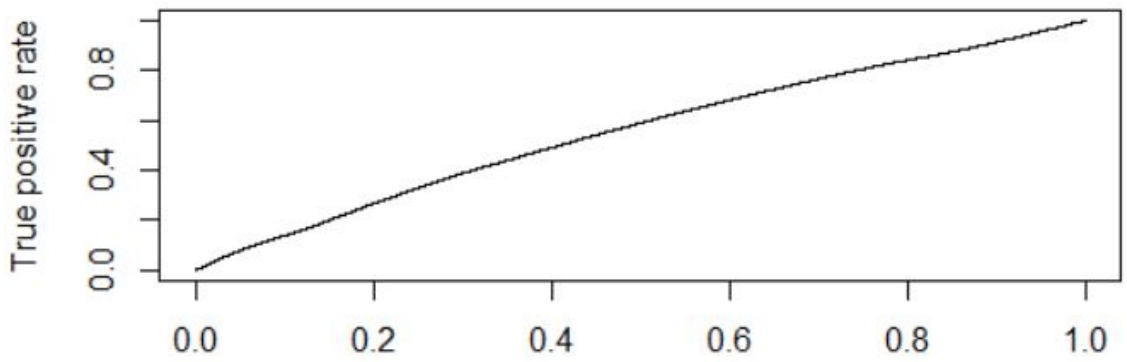
We would definitely notify anyone who uses our model of how we organized the raw data into the data frame that we ultimately ended up performing analysis with. We organized the raw event log data by sessions, so that each session was an “observation” in the design matrix. We also defined the viewer count of each session as the maximum viewer count observed at any given point of the session - if one is interested in the behavior of the viewer count within a session, our cleaned dataset would not contain information about that. We may also have issues with multiple hypothesis testing, since we do have around 15 covariates - it would not be anomalous if one or two of these covariates to show up as significant just because of pure chance, not because of their actual underlying link to the viewer count. In addition, we manually chose the covariates in the beginning based on our intuition that this variable would or would not matter - by performing this process, we might be vulnerable to post-selection inference. A more robust analysis would try to account for all these potential roadblocks.

We did not collect the data so we cannot really affect the data collection process, but if we could, we would collect more covariates that may be helpful in building practical models. For example, Overall traffic through the site at a given time would be a reasonable covariate that may help with our analysis, so that we can normalize the “relative popularity,” or what percentage of online viewers were at the site because of this specific session. In addition, we would make data collection more uniform over time. We noticed that information is not collected over uniform intervals of time, which made the data cleaning process a bit confusing.

If we started the project over again, we may change the data cleaning process so that we can capture information about the viewer count within a session over time. The way we did it this time loses all temporal resolution within the session, and it may interesting to see what kind of features makes sessions lose viewers quickly (even if a session started with a high viewer count). We believe we still would have performed similar viewer count type of analysis.

Appendix:

ROC curve for classification on test set (what would have happened if I changed my threshold):




```
#removing NA and empty entries#
by_session = by_session[-which(by_session$uptime == ""), ]
by_session = by_session[-which(is.na(by_session$max_viewer)), ]

cleaned_timestamp = c()
for (i in 1:nrow(by_session)) {
  timestamp = toString(by_session$uptime[i])
  ts = unlist(strsplit(timestamp, "\\s+"))
  cleaned_timestamp[i] = chron(paste(ts[2], ts[3], ts[5]), ts[4], format = c('mon d y', 'h:m:s'))
}

by_session$uptime = cleaned_timestamp

by_session$US = by_session$geo == 'US'
by_session = subset(by_session, select = -c(geo))

by_session$eng_lang = by_session$language_session == 'en'
by_session$channel_subscription = by_session$channel_subscription == 'True'
by_session$featured = by_session$featured == 'True'
by_session$mature = by_session$mature == 'True'
```

```
by_session = subset(by_session, select = -c(language_session, broadcaster))

by_session.reg = subset(by_session, select = -c(session_id, channel_id, language_channel,
timezone, producer))
by_session.reg = as.data.frame(data.matrix(by_session.reg))
by_session.reg$uptime = by_session.reg$uptime %% 1
fm = lm(max_viewer ~ ., data = by_session.reg)
fm$coefficients

# Bootstrap to generate CI #
# bootstrap 5000 times #
coef_bootstrap = matrix( rep( 0, 10*5000), nrow = 5000, ncol = 10)
for (i in 1:5000) {
  bs_sample = by_session.reg[sample(nrow(by_session.reg), size = nrow(by_session.reg), replace
= TRUE),]
  fm = lm(max_viewer ~ ., data = bs_sample)
  coef_bootstrap[i,] = fm$coefficients
}

library(chron)
library(ROCR)

#by_session$uptime = cleaned_timestamp
classification_data = by_session
classification_data$mature = as.logical(classification_data$mature)*1
classification_data$producer = as.logical(classification_data$producer)*1
classification_data$channel_subscription =
as.logical(classification_data$channel_subscription)*1
classification_data$featured = as.logical(classification_data$featured)*1
classification_data$viewed = (classification_data$max_viewer > 3)*1

# Fit best model 1
fm_model_1 = glm(primetime ~ max_viewer + uptime_sec + channel_view_count + featured +
mature + max_viewer:channel_view_count + mature:featured + max_viewer:uptime_sec,
family='binomial',data=classification_data)

# Test on test data set

# Process the test data set a bit
primetime_test = vector(length=nrow(by_session_test))
for (i in 1:nrow(by_session_test)) {
  timestamp = by_session_test$uptime[i]
  ts = unlist(strsplit(timestamp, "\\s+"))
```

```
  time = strsplit(ts[4], ':')
  if (as.numeric(time[[1]][1]) > 18 && as.numeric(time[[1]][1]) < 22) {
    primetime_test[i] = 1;
  }
}

by_session_test$primetime = primetime_test
test_data = by_session_test
test_data$mature = as.logical(test_data$mature)*1
test_data$producer = as.logical(test_data$producer)*1
test_data$channel_subscription = as.logical(test_data$channel_subscription)*1
test_data$featured = as.logical(test_data$featured)*1
test_data$viewed = (test_data$max_viewer > 3)*1


# Predict on test data
prob = predict(fm_model_1, test_data, type='response')
preds = (prob > 0.15)*1
correct = 0
total = 0
for (i in 1:nrow(by_session)){
  if (!is.na(preds[i]) && !is.na(test_data$primetime[i])){
    total = total + 1
    if (preds[i] == test_data$primetime[i]) {
      correct = correct + 1
    }
  }
}
accuracy = correct/total
# Accuracy:
print(accuracy)

# Get ROC curve and AUC
pred = prediction(prob, test_data$primetime)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf)
auc <- performance(pred, measure = "auc")
auc <- auc@y.values[[1]]
auc
```