435 Project

Crystal Mountain Staffing Model

Kurt Geiger, Kellan Toman, Ryan Lattanzi

November 12, 2018

Abstract

Ticket sales for Crystal Mountain Ski Resort ranging from the years 2013 to 2017 is analyzed, along with several statistical methods developed to predict employment levels required by the company. An ensemble method which combines k-Nearest Neighbors, Poisson, Ordinal, and Linear Regression techniques is developed. We find an accuracy level of 75%, a nearly 30% increase in predictability relative to prior methods.

Introduction

During the course of the past 5 years, our teammate Kurt Geiger has worked for Crystal Mountain Ski Resort, which is a popular skiing and snowboarding destination in Western Washington. Despite Managers generally having a good idea on the level of visits, and hence the level of employment needed to compensate for said visits, a reliable model to predict necessary employment levels is still desired. Therefore our primary objective for this project is to create reliable statistical model(s) to predict employment level (ranked 1-3). A secondary objective is to forecast ticket sales, which go hand-in-hand with employment levels. Our approach first consists of four distinct modelling approaches: first, a Poisson Regression, followed by Ordinal Regression, Linear Regression, and a generalized form of k-Nearest Neighbors. After first summarizing the body of work that went into creating our dataset from what was given, we summarize our modelling approaches in sections 1-3. Our last model is the most complicated, in which we attempted to build an Ensemble of our models in hopes to bolster our predictive power. In the last section, we briefly describe prior methods used by Crystal Mountain Ski Resort management, and compare our model performance to theirs.

Data Collection and Variable Engineering

Our data set came from Crystal Mountain Inc. where Kurt is currently doing an internship. One of the projects He is working on is a staffing projection model that predicts the level of employment needed for any given day during the ski season. The data set included the date, the day of the week, and total daily visits or skiers. Kurt's internship mentor suggested 3 levels of employment, low, medium, and all hands-on deck. He also suggested less than 500 visits would indicate "low" employment, less than 1000 would indicate "medium" employment and anything greater than 1000 would be "high" or all hands-on deck. We created a "Class" variable that directly classifies these quantities of visits into 1="low", 2="medium", and 3="high". We later changed this classification to offer our models more flexibility to less than 1000 equals "low", 1000-2000 equals medium, and any above 2000 to be "high". These classifications are arbitrary and can be adjusted once further research is conducted on how well each department performed during different quantities of visitors.

We created the "Year" variable to keep the seasons separate, which made training and testing more realistic as each season is relatively independent of the other seasons. Christmas break is usually the busiest time of the year in terms of customers, so we created two variables to capture these times. The first was "xmas", a dummy variable indicating whether or not WSU or UW was on winter break (public schools also fell during this timeframe). The second variable we created was "days from christmas" which counts down from the start of the season to Christmas day (which equals zero), then counts back up to the end of the season. The idea was to capture a time reference that represented the closer we get to Christmas, the more visitors we

will have. In this case, we expect the variable to have a negative coefficient. When we are 40 days away from Christmas we expect less visits then when we are a few days away from Christmas.

Another observed peak in visitors comes during spring break, when many students are off from school and get an opportunity to go skiing. We created 3 dummy variables to represent this timeframe, but only one should be used per model. "springb" represents when WSU and UW are on spring break, "springb2" represents when Seattle public schools and other public schools are on spring break, and "sb" represents the timeframe when WSU, UW, and Seattle public schools are on break. Furthermore we created a "break" dummy variable that represents the combination of "xmas" and "springb". From experience, days in which public schools are off due to holiday usually see larger numbers of visitors. "holiday" is a dummy variable representing Thanksgiving, New Years Day, Martin Luther King Jr. Day, and Presidents Day. All other holidays are not during the ski season. Christmas was also not included in this variable because it is already represented in "xmas" and "days from Christmas".

When there is new snow or conditions are good, we usually see an increase in visitors. We found historic snowfall data for Crystal Mountain at onthesnow.com. Our "snowfall" variable represents snowfall in inches. "snowclass" classifies the amount of snowfall into six categories, none, low, mediumlow, medium, mediumhigh, and high. We found that there were too many classifications so we created another variable called "Snowclass" which had 4 classes, none, low, medium, and high. We also created a numeric snowclass, "numsnowclass", but was unused in our models.

The last set of variables we used represented the day of the week. We generally see large crowds during the weekends and small crowds during weekdays. This is due to school and work schedules. First, we created dummy variables for each day of the week, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday. We were worried that our models might find little difference between the mid-week variables Monday-Thursday (Friday usually has slightly larger crowds due to people either skipping work on Friday or working four days a week). Also, when using dummy variables, one of the days would be represented in beta knot as a basis function. So, we created multiple midweek dummy variables to play around with. "weekday" represents Monday-Thursday, "wkday" represents Monday-Wednesday, and "wkday2" represents Tuesday-Thursday.

Approach One: Poisson Regression

The rationale behind running a Poisson Regression is that we will be predicting counts, which is precisely what a Poisson Regression sets out to do. Since our ultimate goal is to predict classes 1, 2, or 3, we will first predict total visits and place them into corresponding classes.
Before implementing Poisson Regression, we will first perform feature selection so that our model has the least predictors to make an effect.

```
library(mlbench)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
data = read.csv("Crystalr.csv")
data$Date <- NULL     # Don't need date
data$Day.of.Week = NULL  # Already used in dummy variables
data$snowclass = NULL # Need features to be numeric for caret package
cl = data$Class # Extract the class column
```

```
data$Class = NULL

cormat = cor(data[,2:16])
highcor = findCorrelation(cormat, cutoff=0.75)
highcor
```

```
## [1] 5 8
```

The correlation matrix tells us that features 5 and 8 have a correlation higher that 0.75 with other features, so we remove them. They correspond to features "spring" and "snowfall". Spring is a boolean feature that indicates whether it is spring break at WSU or UW, and snowfall includes inches of snowfall that day. Spring is probably correlated with the column "break", which is boolean that indicates whether WSU or UW is either on spring or winter break. Snowfall is definitely correlated with the column "numsnowclass" that places the inches of snow fallen into classes 0-5.

```
data$spring = NULL
data$snowfall = NULL

m = mean(c(data$Visits))
v = var(c(data$Visits))
m
```

```
## [1] 2023.609
```

```
v
```

```
## [1] 3111541
```

The huge variance suggests we do not have the Poisson condition that mean = var. Hence, we have overdispersion. In this case the MLE is okay, but the SE are wrong. In order to account for this, we will implement a quasipoisson regression.

Now, we split the data into training and testing sets: training is the first two years and testing is the last year.

```
year1 = data[data$Year == 1,]
year2 = data[data$Year == 2,]
train = rbind(year1,year2)
train = train[,-2]
test = data[data$Year == 3,]
test = test[,-2]
```

Now, we fit our model:

```
fit = glm(Visits ~ ., family = quasipoisson, data = train)
summary(fit)
```

```
##
## Call:
## glm(formula = Visits ~ ., family = quasipoisson, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -66.266  -19.207   -4.423    9.665  101.890
##
## Coefficients: (2 not defined because of singularities)
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         7.934931   0.121638  65.234  < 2e-16 ***
## xmas                0.207751   0.188164   1.104   0.2705
```

```
## days.from.christmas -0.003527   0.001515  -2.328   0.0206 *
## break.               0.132838   0.146153   0.909   0.3642
## holiday              0.933270   0.195595   4.771 2.93e-06 ***
## numsnowclass         0.116085   0.022394   5.184 4.14e-07 ***
## weekday             -0.939620   0.141864  -6.623 1.76e-10 ***
## Friday              -0.537967   0.126713  -4.246 2.96e-05 ***
## Saturday             0.238432   0.099785   2.389   0.0175 *
## Sunday                    NA         NA      NA       NA
## Monday              -0.104997   0.174318  -0.602   0.5474
## Tuesday             -0.107813   0.177959  -0.606   0.5451
## Wednesday           -0.013486   0.171177  -0.079   0.9373
## Thursday                  NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 737.2465)
##
##     Null deviance: 401246  on 294  degrees of freedom
## Residual deviance: 189578  on 283  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

We will get rid of insignificant predictors, and fit a new model that will show all predictors as significant.

```
rid = c(10:14)
train = train[,-c(rid,2,4)]
test = test[,-c(rid,2,4)]

fit2 = glm(Visits ~ ., family = quasipoisson, data = train)
summary(fit2)
```

```
##
## Call:
## glm(formula = Visits ~ ., family = quasipoisson, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -53.918  -18.857   -4.114   10.018  112.044
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          8.152146   0.100262  81.308  < 2e-16 ***
## days.from.christmas -0.005902   0.001159  -5.094 6.36e-07 ***
## holiday              0.913546   0.198676   4.598 6.39e-06 ***
## numsnowclass         0.108638   0.022937   4.736 3.42e-06 ***
## weekday             -1.013240   0.100739 -10.058  < 2e-16 ***
## Friday              -0.557004   0.130193  -4.278 2.57e-05 ***
## Saturday             0.236374   0.102965   2.296   0.0224 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 784.7386)
##
##     Null deviance: 401246  on 294  degrees of freedom
```

```
## Residual deviance: 198958  on 288  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

Using this new model, we will predict visits on our test set, and place them corresponding classes

```
pred = as.numeric(predict(fit2, newdata = test, type = "response"))

pred = floor(pred) # Change them to integers
cl = rep(0,length(pred)) # Will give predicted class values
cl[pred<1000] = 1
cl[pred>= 1000 & pred<2000] = 2
cl[pred>= 2000] = 3

response = rep(0,length(test[1])) # True class values
response[test$Visits<1000] = 1
response[test$Visits>=1000 & test$Visits<2000] = 2
response[test$Visits>=2000] = 3
```

Compute the accuracy of the model:

```
j = 1
for (i in 1:length(response)){

  if (response[i] == cl[i]){
    j = j + 1
  }
}

acc = j/length(response)
j # Correctly classified test examples
```

```
## [1] 96
```

```
acc
```

```
## [1] 0.64
```

We predicted 96/150 correctly which is 64 %. Let us now compare with a model that predicts all 3's:

```
x = which(response == 3)
length(x)
```

```
## [1] 62
```

```
acc = length(x)/length(response)
acc
```

```
## [1] 0.4133333
```

With this model, we will predict 62/150, or 41 %. Hence, our Poisson model improves upon this model by about 50 %.

Now, it is better to overpredict than underpredict:

```
length(which(cl<response)) # Tells us we underpredicted 25 out of 150 times
```

```
## [1] 25
```

We underpredicted 25 out of 150 times, but we will further figure out by how much we underpredicted:

```
under = c(which(cl<response))
j = test$Visits[under] - pred[under]
length(which(j>=1000))
```

```
## [1] 10
```

Thus, for our underpredictions, we were within 1000 customers 15 out of the 25 times. Although the accuracy is not ideal, we hope it will contribute to the ensemble presented in a later section.

Approach two: Ordinal and Linear Regression

First, we run a linear regression. We split the data into testing and training sets, years 1 & 2 is training, year 3 is testing. We also create a new class for employment levels, with wider margins than the original class.

```
CM <- read.csv("Crystalk.csv")

yr12=which(CM[,4]!=3)
CM12=CM[1:length(yr12),]

y3=which(CM[,4]==3)
CM3=CM[(length(yr12)+1):length(CM[,6]),]

cl = c()
cl[CM12$Visits<1000] = 1
cl[CM12$Visits>=1000 & CM12$Visits<2000] = 2
cl[CM12$Visits>=2000] = 3

cl2 = c()
cl2[CM3$Visits<1000] = 1
cl2[CM3$Visits>=1000 & CM3$Visits<2000] = 2
cl2[CM3$Visits>=2000] = 3
```

Trying different combinations of predictors, we observe the following model had the best predictability.

```
a = lm(Visits~xmas+springb2+days.from.christmas+snowclass+Monday+Tuesday+Wednesday+
       Thursday+Friday, data=CM12)
summary(a)
```

```
##
## Call:
## lm(formula = Visits ~ xmas + springb2 + days.from.christmas +
##     snowclass + Monday + Tuesday + Wednesday + Thursday + Friday,
##     data = CM12)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2834.5  -724.6   -71.2   449.6  5215.2
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          4471.309    359.053  12.453  < 2e-16 ***
## xmas                  967.463    239.247   4.044 6.80e-05 ***
## springb2              902.706    308.393   2.927 0.003701 **
## days.from.christmas    -5.848      2.679  -2.182 0.029908 *
```

```
## snowclassLow        -1225.702    344.809  -3.555 0.000444 ***
## snowclassMedium      -790.679    499.023  -1.584 0.114215
## snowclassMediumHigh  -319.349    448.446  -0.712 0.476978
## snowclassMediumLow   -706.255    408.193  -1.730 0.084692 .
## snowclassNone       -1246.043    329.460  -3.782 0.000190 ***
## Monday              -2194.120    242.759  -9.038  < 2e-16 ***
## Tuesday             -2282.346    244.496  -9.335  < 2e-16 ***
## Wednesday           -2125.229    242.017  -8.781  < 2e-16 ***
## Thursday            -2135.235    240.945  -8.862  < 2e-16 ***
## Friday              -1495.257    242.746  -6.160 2.51e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1276 on 281 degrees of freedom
## Multiple R-squared:  0.4786, Adjusted R-squared:  0.4545
## F-statistic: 19.84 on 13 and 281 DF,  p-value: < 2.2e-16
```

```
P=predict(a,newdata=CM3)

P[P<1000] = 1
P[P>=1000 & P<2000] = 2
P[P>=2000] = 3

lmt = table(P,cl2)
lmacc = (lmt[1]+lmt[5]+lmt[9])/(length(CM3[,6]))
lmt
```

```
##    cl2
## P    1  2  3
##   1 38 10  1
##   2 11 19  7
##   3  1  9 54
```

```
lmacc
```

```
## [1] 0.74
```

This model predicted correctly 74 percent of the time. However, with such narrow snow classes, medium low, medium, and medium high predictors were insifnificant.
using the wider margin predictor “Snowclass”, we get.

```
b = lm(Visits~xmas+springb2+days.from.christmas+Snowclass+Monday+Tuesday+Wednesday+
       Thursday+Friday, data=CM12)
summary(b)
```

```
##
## Call:
## lm(formula = Visits ~ xmas + springb2 + days.from.christmas +
##     Snowclass + Monday + Tuesday + Wednesday + Thursday + Friday,
##     data = CM12)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2828.6  -738.3   -74.4   464.0  5216.6
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)          4318.355    288.518  14.967  < 2e-16 ***
## xmas                  962.512    237.883   4.046 6.72e-05 ***
## springb2              891.760    307.098   2.904  0.00398 **
## days.from.christmas    -5.754      2.653  -2.168  0.03096 *
## Snowclasslow        -1073.987    269.292  -3.988 8.48e-05 ***
## Snowclassmedium      -582.143    314.663  -1.850  0.06535 .
## Snowclassnone       -1092.610    247.836  -4.409 1.48e-05 ***
## Monday              -2198.230    241.999  -9.084  < 2e-16 ***
## Tuesday             -2277.421    241.995  -9.411  < 2e-16 ***
## Wednesday           -2136.190    240.929  -8.866  < 2e-16 ***
## Thursday            -2144.881    239.420  -8.959  < 2e-16 ***
## Friday              -1495.463    241.986  -6.180 2.23e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1272 on 283 degrees of freedom
## Multiple R-squared:  0.4776, Adjusted R-squared:  0.4573
## F-statistic: 23.52 on 11 and 283 DF,  p-value: < 2.2e-16
```

```
P=predict(b,newdata=CM3)

P[P<1000] = 1
P[P>=1000 & P<2000] = 2
P[P>=2000] = 3

lmt = table(P,cl2)
lmacc = (lmt[1]+lmt[5]+lmt[9])/(length(CM3[,6]))
lmt
```

```
##    cl2
## P    1  2  3
##   1 38 10  1
##   2 12 19  8
##   3  0  9 53
```

```
lmacc
```

```
## [1] 0.7333333
```

Here we sacrifice a small amount of accuracy, but achieve more significant predictors. It should be noted the main difference is the previous model predicted a 3 when it was actually a 1, where iin this model it predicted a 2 when it was actually a 3. In terms of staffing, it would be better to predict 1 class away rather than 2. So the second model is preffered in this regard.

Next we run an Ordinal Regression. Also known as Ordinal Classification or Ordinal Logistic Regression, it is used for predicting a variable whose value exists on an arbitrary scale where only the relative ordering is significant. In this case it's 1,2 or 3 for low, medium, and high levels of employment.

```
library(foreign)
library(ggplot2)
library(MASS)
library(Hmisc)
```

```
## Loading required package: survival
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(reshape2)
```

Using original class, the following model HAD 72% prediction accuracy. As you can see this models performance decreased to 68% accuracy. It performed when class 3 was any visits over 1000, because 64% was the number of true class 3's.

```
m<- polr(as.factor(cl)~xmas+holiday+Snowclass+
           Monday+Tuesday+Wednesday+Thursday+Friday,
         data=CM12, Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = as.factor(cl) ~ xmas + holiday + Snowclass + Monday +
##     Tuesday + Wednesday + Thursday + Friday, data = CM12, Hess = TRUE)
##
## Coefficients:
##                    Value Std. Error t value
## xmas              2.0239     0.3732   5.423
## holiday           3.3737     0.9952   3.390
## Snowclasslow     -1.6200     0.4622  -3.505
## Snowclassmedium -0.8516     0.5420  -1.571
## Snowclassnone    -1.8681     0.4357  -4.287
## Monday           -3.7941     0.4785  -7.930
## Tuesday          -3.9310     0.4956  -7.932
## Wednesday        -3.2632     0.4486  -7.275
## Thursday         -3.1740     0.4363  -7.275
## Friday           -1.6265     0.3937  -4.132
##
## Intercepts:
##     Value   Std. Error t value
## 1|2 -3.9649  0.5153    -7.6940
## 2|3 -2.1241  0.4643    -4.5747
##
## Residual Deviance: 450.2428
## AIC: 474.2428
```

```
p=predict(m,newdata=CM3)

polyrt = table(p,CM3$cl)
polyracc = (polyrt[1]+polyrt[5]+polyrt[9])/(length(cl2))
polyrt
```

```
##
## p    1  2  3
##   1 47 18  5
```

```
##   2  3 11 12
##   3  0  9 45
```

```
polyracc
```

```
## [1] 0.6866667
```

Here, the coefficients are as we predicted. with no snow fewer visits occur, as you get into the low, then medium classes, fewer visits are deducted. Also monday-thursday has similar coefficients, where friday is about half the size, suggesting more visits occur on fridays then during midweek, which we've observed to be true. Also, christmas and holidays have a significant positive influence on ticket sales or visits.

Our next ordinal regression is

```
m<- polr(as.factor(cl)~xmas+sb+holiday+snowclass+
           Monday+Tuesday+Wednesday+Thursday+Friday,
         data=CM12, Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = as.factor(cl) ~ xmas + sb + holiday + snowclass +
##     Monday + Tuesday + Wednesday + Thursday + Friday, data = CM12,
##     Hess = TRUE)
##
## Coefficients:
##                        Value Std. Error t value
## xmas                  2.2971     0.3975   5.778
## sb                    0.5411     0.3007   1.799
## holiday               3.6342     1.0233   3.552
## snowclassLow         -2.4021     0.6495  -3.698
## snowclassMedium      -1.0462     0.8515  -1.229
## snowclassMediumHigh -1.2300     0.7814  -1.574
## snowclassMediumLow  -1.8221     0.7469  -2.440
## snowclassNone        -2.5768     0.6295  -4.093
## Monday               -3.9130     0.4912  -7.967
## Tuesday              -4.1028     0.5185  -7.912
## Wednesday            -3.2296     0.4500  -7.177
## Thursday             -3.2239     0.4410  -7.310
## Friday               -1.6540     0.3947  -4.190
##
## Intercepts:
##     Value   Std. Error t value
## 1|2 -4.5404  0.6985    -6.5001
## 2|3 -2.6666  0.6539    -4.0783
##
## Residual Deviance: 443.4606
## AIC: 473.4606
```

```
p=predict(m,newdata=CM3)

polyrt = table(p,CM3$cl)
polyracc = (polyrt[1]+polyrt[5]+polyrt[9])/(length(cl2))
polyrt
```

```
##
## p    1  2  3
```

```
##   1 48 20  5
##   2  2  9 10
##   3  0  9 47
```

```
polyracc
```

```
## [1] 0.6933333
```

With this model we see a slight improvement in predictabilty, But there are some small t-values/higher p-values occur. Specifically medium, mediumhigh snow classifications and spring break is slightly short of being significant.

The following model offers one of the highest predictabilities at 70%, but the wkday variable (mon-wed), is not very significant with a t-value of -1.25. Removing that predictor we get a 69.33% accuracy rate. It should be noted how the numsnowclass variable works.It was designed to be a factor, but without using a factor we get a similar effect. As more snow falls, a higher quantity of visitors come to crystal mountain.

```
m<- polr(as.factor(cl)~xmas+holiday+numsnowclass+
           Friday+Saturday+Sunday,
         data=CM12, Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = as.factor(cl) ~ xmas + holiday + numsnowclass +
##     Friday + Saturday + Sunday, data = CM12, Hess = TRUE)
##
## Coefficients:
##               Value Std. Error t value
## xmas         2.0259    0.36753   5.512
## holiday      3.2482    0.93936   3.458
## numsnowclass 0.4396    0.09065   4.850
## Friday       1.9279    0.35590   5.417
## Saturday     4.0695    0.46572   8.738
## Sunday       3.0964    0.40378   7.668
##
## Intercepts:
##     Value   Std. Error t value
## 1|2  1.4976  0.2299     6.5138
## 2|3  3.3473  0.3034    11.0325
##
## Residual Deviance: 448.5954
## AIC: 464.5954
```

```
p=predict(m,newdata=CM3)

polyrt = table(p,CM3$cl)
polyracc = (polyrt[1]+polyrt[5]+polyrt[9])/(length(cl2))
polyrt
```

```
##
## p    1  2  3
##   1 47 18  3
##   2  3 11 13
##   3  0  9 46
```

```
polyracc
```

```
## [1] 0.6933333
```

running the same model as above with "numsnowclass" as a factor we obtain a prediction accuracy rate of 70.66%. Only the factor of snowclass 1 is insignificant.

```
m<- polr(as.factor(cl)~xmas+holiday+as.factor(numsnowclass)+
           Friday+Saturday+Sunday,
         data=CM12, Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = as.factor(cl) ~ xmas + holiday + as.factor(numsnowclass) +
##     Friday + Saturday + Sunday, data = CM12, Hess = TRUE)
##
## Coefficients:
##                             Value Std. Error t value
## xmas                       2.0676     0.3738  5.5310
## holiday                    3.2472     0.9407  3.4518
## as.factor(numsnowclass)1 0.2240     0.3074  0.7288
## as.factor(numsnowclass)2 0.8604     0.4790  1.7962
## as.factor(numsnowclass)3 1.3234     0.6504  2.0349
## as.factor(numsnowclass)4 1.3754     0.5495  2.5028
## as.factor(numsnowclass)5 2.5204     0.6133  4.1099
## Friday                     1.9304     0.3585  5.3840
## Saturday                   4.0907     0.4659  8.7804
## Sunday                     3.0870     0.4068  7.5888
##
## Intercepts:
##     Value   Std. Error t value
## 1|2  1.4351  0.2470     5.8106
## 2|3  3.2911  0.3151    10.4449
##
## Residual Deviance: 447.2173
## AIC: 471.2173
```

```
p=predict(m,newdata=CM3)

polyrt = table(p,CM3$cl)
polyracc = (polyrt[1]+polyrt[5]+polyrt[9])/(length(cl2))
polyrt
```

```
##
## p    1  2  3
##   1 47 18  3
##   2  3 11 11
##   3  0  9 48
```

```
polyracc
```

```
## [1] 0.7066667
```

This last model is my preferred model. It had the fewest (3) bad errors, predicting a 1 when staffing level should be 3.

Approach Three: kNN With Alternative Metric

In this section, we attempt to develop a kNN model by engineering a metric that makes sense for the problem. Our hope is that the kNN model will be able to capture not only temporal patterns in the dataset, but also fill in gaps in prediction power relative to the prior three models mentioned.

After considerable consideration, we landed on using the following metric:

$$d(x, y) = |day_x - day_y| + \lambda|t_x - t_y|$$

Where $\lambda \in R^+$ is a scaling value in order to scale the time difference to be proportionate to the difference in ranked days. To be more precise, the variable *day* was defined as the relative ranks of each day of the week; Monday corresponds to a 1, while Saturday corresponds to a 7:

```
data=read.csv("Crystalr.csv")
yr12=which(data[,4]!=3)
data12=data[1:length(yr12),]
```

```
Rday=vector(length=length(data[,6]))
for(i in 1:length(data[,6]))
{
    if (data[i,18]==1)
    {
        Rday[i]=1
    }
    else if (data[i,19]==1)
    {
        Rday[i]=2
    }
    else if (data[i,20]==1)
    {
        Rday[i]=3
    }
    else if (data[i,21]==1)
    {
        Rday[i]=4
    }
    else if (data[i,17]==1)
    {
        Rday[i]=6
    }
    else if (data[i,16]==1)
    {
        Rday[i]=7
    }
    else
    {
        Rday[i]=5
```

```
    }
}
```

These ranks were predetermined by Kurt's ex-employer, manager at Crystal Mountain Ski Resort. Next, the time-variable t corresponds to a simple variable that ranges from 1 to 445 by increments of 1, where 445 denotes the length of our entire dataset.

```
t=seq(1,445,1)
```

Theorem: d(x,y) defines a proper metric

Prf:
For the proof, we will break up the metric into two separate pieces denoted $d_1(x, y)$ and $d_2(x, y)$, where:

$$d_1(x, y) = |day_x - day_y|$$

$$d_2(x, y) = \lambda|t_x - t_y|$$

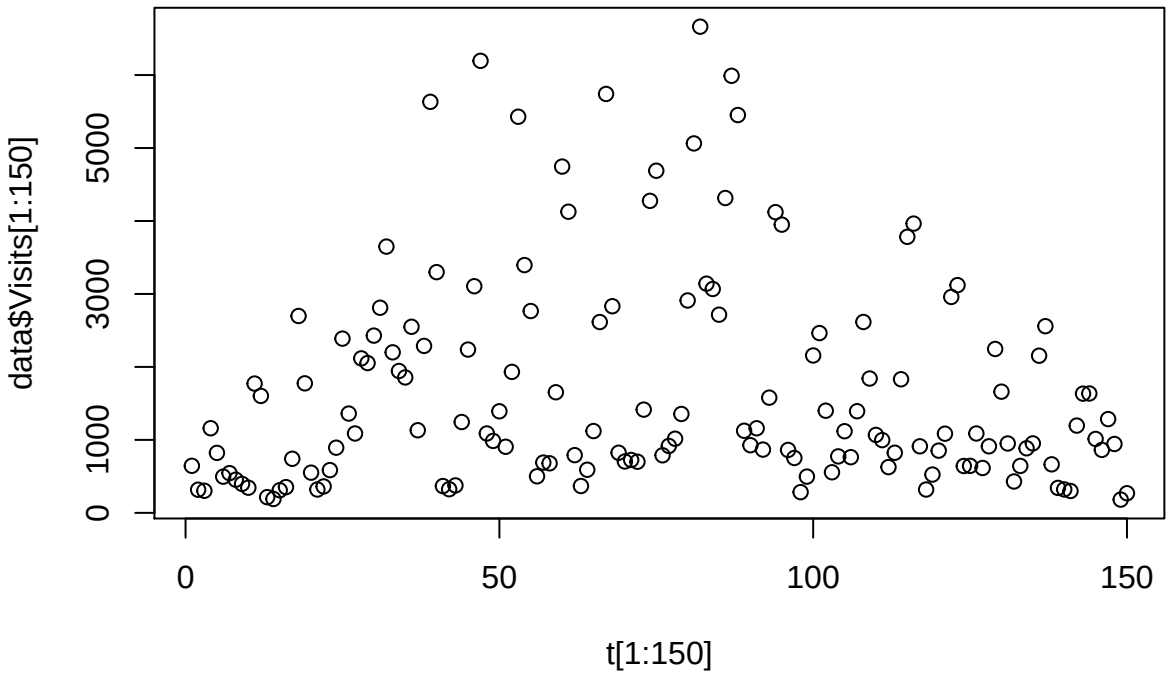
$d_1(x, y)$ is trivially a metric:

- $d_1(x, y) \geq 0$
- $d_1(x, y) = 0$ iff $x = y$
- $d_1(x, y) = d_1(y, x)$
- $d_1(x, y) \leq d_1(x, y) + d_1(y, z)$

$d_2(x, y)$ is equivalent to $\lambda||_1$, where $||_1$ denotes the usual absolute value metric in R. Since λ is any finite value in R^+, we conclude $d_2(x, y)$ is a metric. Finally, for any two metrics $d_1(x, y)$ and $d_2(x, y)$, it's widely known that $d_1(x, y) + d_2(x, y)$ is a metric.

The purpose for defining this metric is two-fold: First, it's known that ticket sales (and thus employment rate) at Crystal Mountain Ski Resort has seasonally varying parameters, as can be justified by the following plot:

```
plot(t[1:150],data$Visits[1:150])
```



By defining part of our distance to be the difference in time, we hope that our kNN can capture a part of this seasonal variance. Secondly, ticket sales are highly dependent on day of the week; for example, a Saturday tends to have much higher ticket sales than a Monday. Therefore, adding as part of our metric the difference in ranked-day of the week may capture this pattern. Finally, with the simplicity of this approach, we hope that we can capture temporally varying parameters more efficiently than the prior models, which may be beneficial to the Ensemble model in the next section.

The logic behind creating our kNN algorithm is simple: first, we define a distance matrix (dependent on λ), followed by taking the mean of the first k nearest points as defined by the metric to the point we wish to predict:

```
Dknn=function(L,d)
{
    Rday=Rday[1:d]
    t=t[1:d]
    D=matrix(0,(d),(d))
    for(i in 1:d)
    {
        for(j in 1:d)
        {
            D[i,j]=abs(Rday[i]-Rday[j]) + L*abs(t[i]-t[j])
        }
    D[i,]=order(D[i,])
    }

    return(D)
}

gknn=function(L,k,d)
{
    pred=vector(length=d)
    D=Dknn(L,d)
```

```
    for (i in 1:d)
    {
        pred[i]=mean(truec[D[i,2:(k+1)]])
        pred[i]=round(pred[i])
    }
return(pred)
}
```

The next, more challenging part is to run a cross-validation over our dataset in order to find the optimal parameter λ. An important piece to note here is how we chose to do cross-validation: we do not conduct simple random sampling of our entire dataset. Due to the complex seasonal dependencies across the three years in our dataset, we chose to fit each model on our first year's data (2014-2015) and validate it on the subsequent year (2015-2016). Finally, we can see how well the model predicts the employment levels for the last year (2016-2017). We find the following tables and predictive accuracy estimates:

```
##      [,1]      [,2]
## [1,]   51 0.6516854
## [2,]   18 0.6202247
## [3,]   51 0.7078652
## [4,]   41 0.6516854
## [5,]   27 0.6853933
## [6,]   16 0.6404494
## [7,]   21 0.6674157
## [8,]   18 0.6359551
```

From Cross Validating over both the variables k and λ, we find our best performing model is when $k = 3$ and $\lambda = .51$. The following is it's predictive accuracy over the 3rd year:

```
b=gknn(.51,3,445)
c=table(b[296:445],cl2)
table(b[296:445],cl2)
```

```
##    cl2
##      1  2  3
##   1 33  5  1
##   2 17 28 24
##   3  0  5 37
```

```
(c[1]+c[5]+c[9])/150
```

```
## [1] 0.6533333
```

A 65.33% accuracy rate is achieved, which was honestly better than we had expected.

Ensemble Method

Our Ensemble method, despite being simple theoretically, was a true nightmare to learn how to code. In the end, we developed an ensemble model that follows the following guideline:
Step 1: Create a partition matrix for all possible combinations predictions made by our selected 5 models

(two kNN models, Poisson Regression, Ordinal Regression, Linear Regression.)

```
#Creating Unique Combinations of predicted Classes
a=matrix(0,3,1)
a[1,1]=1
a[2,1]=2
a[3,1]=3
xx2=c(a,a,a,a,a)
x2=combn(xx2,5)
x2=t(x2)
x2=unique(x2)
s1=function(m1t,m2t,m3t,m4t,m5t,ytrain)
{
yy=cbind(m1t,m2t,m3t,m4t,m5t,ytrain)

#Organizing Ensemble by Combinations

cc=list()
for(i in 1:length(x2[,1]))
{
  cc[[i]]=as.matrix(yy[((yy[,1]==x2[i,1]) & (yy[,2]==x2[i,2])
                        & (yy[,3]==x2[i,3]) & (yy[,4]==x2[i,4])
                        & (yy[,5]==x2[i,5])),])
}
}
```

Step 2: This partition matrix allows us to find all combinations of our model predictions, so that we may find our prediction by the following criteria:

$$\hat{y} = argmax_{P(y_j)}[P(y_1|x_{1_i}, x_{2_i}x_{3_i}x_{4_i}x_{5_i}), P(y_2|x_{1_i}, x_{2_i}x_{3_i}x_{4_i}x_{5_i}), P(y_2|x_{1_i}, x_{2_i}x_{3_i}x_{4_i}x_{5_i})]$$

Where $x_{1...5_i}$ denote our five model predictions for each observation of the class y_j.

```
#Fitting Ensemble
fitens=function(cc)
{
type=c()
acc=c()
for(i in 1:length(x2[,1]))
{
    if(length(cc[[i]])>6)
    {
        type[1]=sum(cc[[i]][,6][cc[[i]][,6]==1])/length(cc[[i]][,6])
        type[2]=sum(cc[[i]][,6][cc[[i]][,6]==2])/(2*length(cc[[i]][,6]))
        type[3]=sum(cc[[i]][,6][cc[[i]][,6]==3])/(3*length(cc[[i]][,6]))
        acc[i]=which.max(type)
    }
    else
    {
```

```
        acc[i]=2
    }
}
}
```

Step 3: Finally, we re-organize the incoming data to be predicted, and output our predictions based on the five models:

```
  #Organizing New Data
new=function(m1p,m2p,m3p,m4p,m5p,truetest)
{
  yy2=cbind(m1p,m2p,m3p,m4p,m5p,truetest)
  cc2=list()
  true=c()
  for(i in 1:length(x2[,1]))
  {
    cc2[[i]]=as.matrix(yy2[((yy2[,1]==x2[i,1]) & (yy2[,2]==x2[i,2])
                            & (yy2[,3]==x2[i,3]) & (yy2[,4]==x2[i,4])
                            & (yy2[,5]==x2[i,5])),])
  }
  #Extracting true Y's with correct ordering

  for(i in 1:length(x2[,1]))
  {
    if(length(cc2[[i]])==6)
    {
      cc2[[i]]=t(cc2[[i]])
    }
  }

  #Predicting with Ensemble
  true=list()
  true2=c()
  pred=list()
  for(i in 1:length(x2[,1]))
  {
    if(length(cc2[[i]])==6)
    {
      pred[[i]]=acc[i]
      true[[i]]=cc2[[i]][,6]
    }
    else if (length(cc2[[i]])!=0)
    {
      f=length(cc2[[i]][(cc2[[i]][,1]==x2[i,1]) & (cc2[[i]][,2]==x2[i,2])
                        & (cc2[[i]][,3]==x2[i,3]) & (cc2[[i]][,4]==x2[i,4])
                        & (cc2[[i]][,5]==x2[i,5]),1])
      pred[[i]]=matrix(acc[i],f,1)
      true[[i]]=cc2[[i]][,6]
    }

  }
```

```
  }
```

Putting it all together into one function, our output gives the following confusion matrix and prediction accuracy on the 3rd year:

```
data=read.csv("Crystalr.csv")
CM=read.csv("Crystalk.csv")


#kNN
truec = c()
truec[data$Visits<1000] = 1
truec[data$Visits>=1000 & data$Visits<2000] = 2
truec[data$Visits>=2000] = 3

Rday=vector(length=length(data[,6]))
for(i in 1:length(data[,6]))
{
  if (data[i,18]==1)
  {
    Rday[i]=1
  }
  else if (data[i,19]==1)
  {
    Rday[i]=2
  }
  else if (data[i,20]==1)
  {
    Rday[i]=3
  }
  else if (data[i,21]==1)
  {
    Rday[i]=4
  }
  else if (data[i,17]==1)
  {
    Rday[i]=6
  }
  else if (data[i,16]==1)
  {
    Rday[i]=7
  }
  else
  {
    Rday[i]=5
  }
}
t=seq(1,445,1)

Dknn=function(L,d)
{
```

```
  Rday=Rday[1:d]
  t=t[1:d]
  D=matrix(0,(d),(d))
  for(i in 1:d)
  {
    for(j in 1:d)
    {
      D[i,j]=abs(Rday[i]-Rday[j]) + L*abs(t[i]-t[j])
    }
    D[i,]=order(D[i,])
  }

  return(D)
}

gknn=function(L,k,d)
{
  pred=vector(length=d)
  D=Dknn(L,d)
  for (i in 1:d)
  {
    pred[i]=mean(truec[D[i,2:(k+1)]])
    pred[i]=round(pred[i])
  }
  return(pred)
}
#gknn(1,1,295)

#Ordinal Reg

ordinalreg = function(CM){

  yr12=which(CM[,4]!=3)
  CM12=CM[1:length(yr12),]
  y3=which(CM[,4]==3)
  CM3=CM[(length(yr12)+1):length(CM[,6]),]
  cl = c()
  cl[CM12$Visits<1000] = 1
  cl[CM12$Visits>=1000 & CM12$Visits<2000] = 2
  cl[CM12$Visits>=2000] = 3

  m<- polr(as.factor(cl)~xmas+sb+holiday+snowclass+
             Monday+Tuesday+Wednesday+Thursday+Friday,
           data=CM12)

  trainp = predict(m)

  p=predict(m,newdata=CM3)

  return(list(model = m,trainp = trainp,pred = p))
}

#Linear Reg
```

```r
linearreg = function(CM){

  yr12=which(CM[,4]!=3)
  CM12=CM[1:length(yr12),]
  y3=which(CM[,4]==3)
  CM3=CM[(length(yr12)+1):length(CM[,6]),]
  cl = c()
  cl[CM12$Visits<1000] = 1
  cl[CM12$Visits>=1000 & CM12$Visits<2000] = 2
  cl[CM12$Visits>=2000] = 3

  a = lm(Visits~xmas+springb2+days.from.christmas+Snowclass+Monday+Tuesday+Wednesday+
           Thursday+Friday, data=CM12)

  trainp = predict(a)
  P=predict(a,newdata=CM3)

  P[P<1000] = 1
  P[P>=1000 & P<2000] = 2
  P[P>=2000] = 3

  trainp[trainp<1000] = 1
  trainp[trainp>=1000 & trainp<2000] = 2
  trainp[trainp>=2000] = 3

  return(list(model = a,trainp = trainp, pred = P))
}

#Poisson Reg

poissonreg = function(data){

  data$Date <- NULL
  data$Day.of.Week = NULL
  data$snowclass = NULL
  data$Class = NULL
  data$spring = NULL
  data$snowfall = NULL
  year1 = data[data$Year == 1,]
  year2 = data[data$Year == 2,]
  train = rbind(year1,year2)
  train = train[,-2]
  test = data[data$Year == 3,]
  test = test[,-2]

  rid = c(10:14)
  train = train[,-c(rid,2,4)]
  test = test[,-c(rid,2,4)]

  fit = glm(Visits ~ ., family = quasipoisson, data = train)

  trainp = as.numeric(predict(fit, type = "response"))
```

```
  pred = as.numeric(predict(fit, newdata = test, type = "response"))

  trainp = floor(trainp)
  pred = floor(pred)

  pred[pred<1000] = 1
  pred[pred>= 1000 & pred<2000] = 2
  pred[pred>= 2000] = 3

  trainp[trainp<1000] = 1
  trainp[trainp>= 1000 & trainp<2000] = 2
  trainp[trainp>= 2000] = 3

  return(list(model = fit, trainp = trainp,pred = pred))
}
# Get each model
ord = ordinalreg(CM)
lin = linearreg(CM)
poi = poissonreg(data)

# Extract training predictions from each model
pord = as.numeric(ord$trainp)
plin = as.numeric(lin$trainp)
ppoi = as.numeric(poi$trainp)
#pknn1 = gknn(.27,5,295)
pknn1 = gknn(3,1,295)
pknn2 = gknn(.01,2,295)
#pknn2 = gknn(.51,1,295)

# Extract testing predictions from each model
predord = as.numeric(ord$pred)
predlin = as.numeric(lin$pred)
predpoi = as.numeric(poi$pred)
#predknn1 = gknn(.27,5,445)[296:445]
predknn1 = gknn(3,1,445)[296:445]
predknn2 = gknn(.01,2,445)[296:445]
#predknn2 = gknn(.51,1,445)[296:445]
# Get the true class values for training set
yr12=which(CM[,4]!=3)
CM12=CM[1:length(yr12),]
truetrain = c()
truetrain[CM12$Visits<1000] = 1
truetrain[CM12$Visits>=1000 & CM12$Visits<2000] = 2
truetrain[CM12$Visits>=2000] = 3

# Get the true class values for testing set
y3=which(CM[,4]==3)
CM3=CM[(length(yr12)+1):length(CM[,6]),]
truetest = c()
truetest[CM3$Visits<1000] = 1
truetest[CM3$Visits>=1000 & CM3$Visits<2000] = 2
truetest[CM3$Visits>=2000] = 3
Ens=function(ytrain,ytrue,m1t,m2t,m3t,m4t,m5t,m1p,m2p,m3p,m4p,m5p)
```

```
{




  #Creating Unique Combinations of predicted Classes
  a=matrix(0,3,1)
  a[1,1]=1
  a[2,1]=2
  a[3,1]=3
  xx2=c(a,a,a,a,a)
  x2=combn(xx2,5)
  x2=t(x2)
  x2=unique(x2)
  yy=cbind(m1t,m2t,m3t,m4t,m5t,ytrain)

  #Organizing Ensemble by Combinations

  cc=list()
  for(i in 1:length(x2[,1]))
  {
    cc[[i]]=as.matrix(yy[((yy[,1]==x2[i,1]) & (yy[,2]==x2[i,2])
                          & (yy[,3]==x2[i,3]) & (yy[,4]==x2[i,4])
                          & (yy[,5]==x2[i,5])),])
  }

  #Fitting Ensemble
  type=c()
  acc=c()
  for(i in 1:length(x2[,1]))
  {
    if(length(cc[[i]])>6)
    {
      type[1]=sum(cc[[i]][,6][cc[[i]][,6]==1])/length(cc[[i]][,6])
      type[2]=sum(cc[[i]][,6][cc[[i]][,6]==2])/(2*length(cc[[i]][,6]))
      type[3]=sum(cc[[i]][,6][cc[[i]][,6]==3])/(3*length(cc[[i]][,6]))
      acc[i]=which.max(type)
    }
    else
    {
      acc[i]=2
    }
  }

  #Organizing New Data
  yy2=cbind(m1p,m2p,m3p,m4p,m5p,truetest)
  cc2=list()
  true=c()
  for(i in 1:length(x2[,1]))
  {
    cc2[[i]]=as.matrix(yy2[((yy2[,1]==x2[i,1]) & (yy2[,2]==x2[i,2])
                            & (yy2[,3]==x2[i,3]) & (yy2[,4]==x2[i,4])
```

```
                            & (yy2[,5]==x2[i,5])),])
  }
  #Extracting true Y's with correct ordering

  for(i in 1:length(x2[,1]))
  {
    if(length(cc2[[i]])==6)
    {
      cc2[[i]]=t(cc2[[i]])
    }
  }

  #Predicting with Ensemble
  true=list()
  true2=c()
  pred=list()
  for(i in 1:length(x2[,1]))
  {
    if(length(cc2[[i]])==6)
    {
      pred[[i]]=acc[i]
      true[[i]]=cc2[[i]][,6]
    }
    else if (length(cc2[[i]])!=0)
    {
      f=length(cc2[[i]][(cc2[[i]][,1]==x2[i,1]) & (cc2[[i]][,2]==x2[i,2])
                        & (cc2[[i]][,3]==x2[i,3]) & (cc2[[i]][,4]==x2[i,4])
                        & (cc2[[i]][,5]==x2[i,5]),1])
      pred[[i]]=matrix(acc[i],f,1)
      true[[i]]=cc2[[i]][,6]
    }

  }
  true2=do.call(c,true)
  true2=data.frame(true2)
  preds=do.call(rbind, pred)
  preds=data.frame(preds)
  hi=cbind(preds,true2)
  a=table(hi[,1],hi[,2])
  (a[1]+a[5]+a[9])/150
  return(list(preds = preds, true2=true2))
}
```

Our Ensemble Model finds a prediciton accuracy of 74.666%, slightly outperforming any of our previous models. The lower than expected performance of our ensemble model may be due to several factors; namely, it is probably overfitting the training sample, as well as having little to no observations for many of the possible combinations of predictions. With a much larger dataset, we may find that this model is much more accurate.

Comparison With Prior Model

Now let us compare these models to the current Crystal Mountain model. Their model uses only day of the week to predict visits. For the testing year (16-17) their model is as follows:

$$y = 27.391x^4 - 449.48x^3 + 2830.6x^2 - 7969.2x + 9509.6$$

where x is day of the week. The model had an $R^2 = .9942$, but it only captured averages of the day of the week, hence missing out on important data such as christmas break and holidays. The predicted average per day were Sunday=3990, Monday=1650, Tuesday=1185, Wednesday=1250, Thursday=1170, Friday=2100, and Saturday=4000. None of these days will fall into the class 1 employment level, all predict more than 1000 visitors. Here we test the accuracy of this model using "cl" where less than 1000 is level 1, 1000-2000 is level 2, and greater than 2000 is level 3 or "high" employment.

```
CM12=matrix(0,1,1)
yr12=which(CM[,4]!=3)
CM12=CM[1:length(yr12),]

CM3=matrix(0,1,1)
y3=which(CM[,4]==3)
CM3=CM[(length(yr12)+1):length(CM[,6]),]

CM3=matrix(0,1,1)
y3=which(CM[,4]==3)
CM3=CM[(length(yr12)+1):length(CM[,6]),]
#table(CM3$spred,CM3$cl)

(0+21+47)/(150)
```

```
## [1] 0.4533333
```

The current model had a prediction accuracy of 45.33%. Part of the problem is that their model never predicts employment class or level 1. This leads to constant overstaffing if used. Again, fitting a polynomial regression over averages does not capture peaks in visitors caused by breaks, holidays or when the snow is good. With our ensemble model having an accuracy of 74.66%, we nearly gain 30% in accuracy.

Future Directions

Despite our Ensemble model performing relatively better than any of the singular models, we believe our ensemble model can be enhancing by attempting a couple different things. Firstly, more data could play a huge role in bolstering its performance. Secondly, we may try to group models pair-wise, and find optimal predictions in clusters of models as opposed to using all five models in one ensemble. Lastly, we could attempt to utilize lagged variables within our dataset in order to gain a better temporal accuracy. In the future, we hope to find a stronger model with a larger dataset.