Building a Model to Predict the Number of Divvy Bikes Rented Per Hour

Divvy bike is Chicago’s bike sharing system. Through this system, riders can rent bikes for limited periods of time and pay for the length of their rental. One of Divvy Bike’s greatest challenges is effectively meeting demand. In order to respond to this, Divvy Bike has to be prepared for spikes in the demand for bikes at stations around the city. Particularly, they have to be able to transport bikes from areas of low demand from areas of high demand. To deal with the entirety of this problem, Divvy would need to build a model of bike returns and bike rentals. In this analysis, I simply create a model of bike rentals, because a simple extension of the model to bike returns would allow Divvy to complete their analysis.

To create this model, I combined data from three separate sources.

1. Divvy Bike’s Data about each rider. <https://www.divvybikes.com/system-data>
2. The Cubs schedule. <https://www.baseball-reference.com/teams/CHC/2016-schedule-scores.shtml>
3. Chicago Weather Data. <https://catalog.data.gov/dataset/beach-weather-stations-automated-sensors>

This model is created as a proof of concept that Divvy can use data about local events and weather throughout Chicago to make more accurate predictions of excess demand.

The datasets were aggregated and combined into 7515 data points consisting of the following fields.

• Checkouts: The number of bikes rented in a given hour in a 1.5 mile radius around Wrigley Field.

• Temp: The temperature in Celsius.

• Humidity: The relative humidity in the air.

• Rain\_Intensity: The Millimeters of rain that fall per hour.

• Rain\_Total: (This dimension was not made clear in the dataset, so it was excluded from analysis)

• Hour: Hour of the day ranging from 0-23.

• DOY: Day of the year ranging from 1 to 365.

• Weekday: The Day of the Week.

• Cubs\_Game: Categorical variable that indicates if a Cubs Game is happening that day at night (N), during the day (D), or not at all/away game (X).

Along with the following engineered fields created during analysis:

• Rain\_Pouring: Categorical variable that takes on the value 1 if the rain intensity is non-zero and takes on the value 0 otherwise.

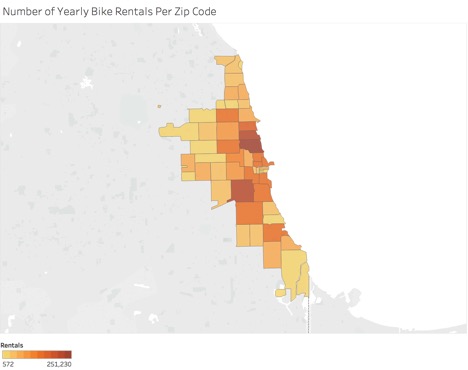
• Game\_Soon: Categorical variable that is 1 if there is a cubs home game happening between hour 10 and 17 (inclusive) if it is a day game or between hour 16 and 23 (inclusive) if it is a night came, and 0 otherwise.

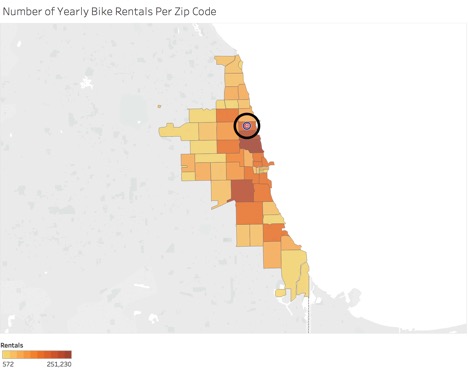
• Weekend: Categorical variable that is 1 if the day of the week is a weekend and 0 otherwise.

This data is recorded for every hour of 2016, but some data points had missing weather values, which is why the total number of points is only 7515.

Overview of the Data:

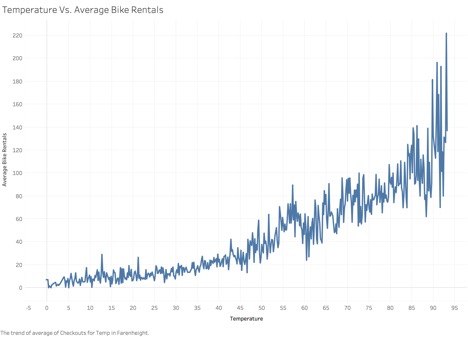
Below are heat maps detailing the number of bikes rented around Chicago. The second map contains the area discussed in this data analysis.





Weather Effects

Temperature was a very important factor in determining the number of bike rentals. It is apparent in the graph below that temperature and the average number of bike rentals have a relatively linear relationship.

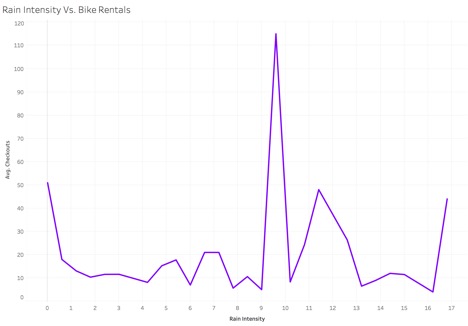


Specifically, for every 1 degree increase in temperature, about 2.5 more bikes are rented per hour, as can be seen in the linear regression.

temp\_lm <- lm(Checkouts ~ Temp, data = wrigley\_agg)  
summary(temp\_lm)

##   
## Call:  
## lm(formula = Checkouts ~ Temp, data = wrigley\_agg)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -88.538 -28.251 -7.894 18.398 254.222   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 17.58886 0.83571 21.05 <2e-16 \*\*\*  
## Temp 2.53249 0.04987 50.78 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 47.27 on 7513 degrees of freedom  
## Multiple R-squared: 0.2555, Adjusted R-squared: 0.2554   
## F-statistic: 2579 on 1 and 7513 DF, p-value: < 2.2e-16

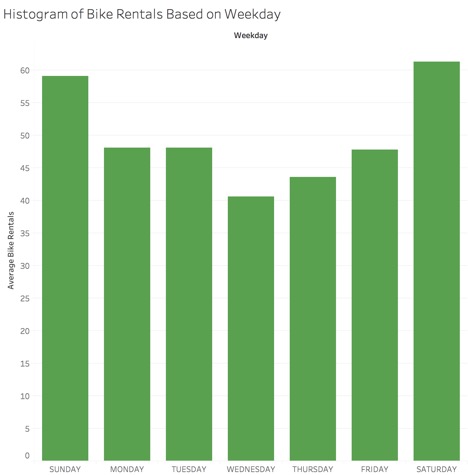
The rain also has a substantial effect upon the number of individuals who decide to rent bikes. In our dataset, there seemed to be a number of outliers that were likely due to some sort of measurement error. The weather data came from Oak Street Beach, which is about 4 miles from Wrigley Field. If there were local variations in the weather, then this could have caused some of the strange effects we see. Below is the graph of rain intensity vs. the number of bike rentals.



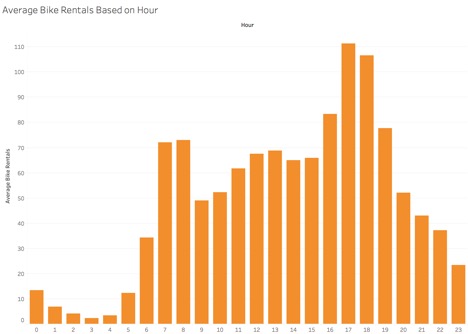
We cannot fairly exclude these outliers, because we don’t know where they come from. However, we can transform the data to just use a variable that determines if it is raining at all to make a prediction. If we look at the average number of bikes rented when it is raining as opposed to when it is not raining, we see a clear trend. About 35 more bikes are rented on average when it is not raining.

Time Effects

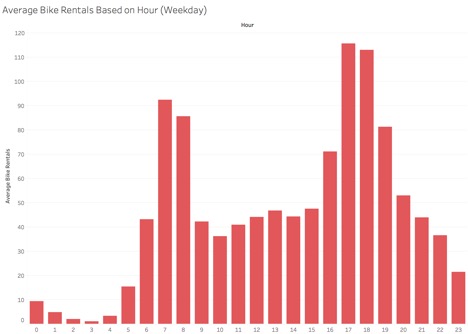
The day of the week is another very important factor in determining the number of riders within an hour. As expected, far more people rent bikes on the weekend than any other day of the week. On an average weekend day, about 15 more bikes are rented than an average weekday. However, there is some variation on weekdays themselves. Specifically, Wednesday and Thursday see substantial drops in the number of bikes that are rented. It is unclear why this is true.



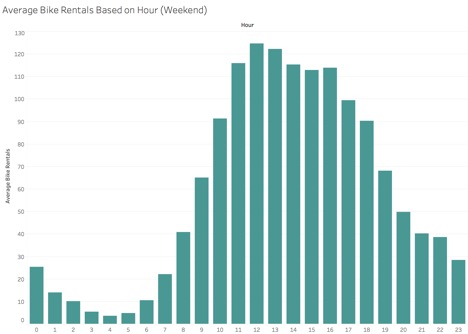
The hour of the day also has a strong influence upon the number of bikes that are rented per hour. However, this effect is a bit more difficult to model because the effect is highly non-linear. The overall distribution of average bike rentals is shown below.



The relationship between the hour of the day and the number of bike checkouts is clearly non-linear. However, the above graph, since it is merely an average over an entire week, does not tell the whole story. The day of the week is very influential upon what hours people rent bikes. The distribution of bike rentals on weekdays is shown below. It is clear that around 7-8 AM and 5-6 PM are the peak hours of bike riding, and that there is a substantial drop in the middle of the day. This can likely be attributed to the fact that most people work on the weekdays.



However, the distribution of bike rentals per hour looks much different on the weekend days. Here, the peak bike rental time are at 12-1 PM and there is no drop in the middle of the day. This interaction between the day of the week and the hour of the day is accounted for in our model.



Cubs Games

A large part of this study is focusing on how to make better predictions for the demand for Divvy Bikes based upon data about local events in the City. In Wrigleyville, a main and recurrent event is Cubs games. We wanted to investigate how much of an effect a Cubs game occurring within an 8 hour window would have upon the demand for Divvy Bikes. Because Cubs games are correlated with nicer weather and more reasonable times of the day for bikers to be riding, it is difficult to isolate the effect of the Cubs playing without a more advanced sampling technique. Instead, we can look at the results of predictive models to see how important cubs games are in predicting bike demand.

Now, it is clear which factors in the data effect the number of bikes that are checked out. The key is to make a model that captures these relationships without overfitting the noise.

The point of the predictive analysis is to build the most accurate predictive model, meansured by 10-fold cross-validated mean-squared error of the number of bikes rented around Wrigley Field in any given hour.

First, we'll try to build a basic linear model to fit the data. Because the data consists of counts, we already know that this is likely not the best fit, because basic linear regression will sometimes predict negative values.

basic\_linear\_1 <- glm(Checkouts ~ Weekday + Game\_Soon + Hour + Temp + Humidity + Rain\_Pouring + DOY, data = wrigley\_agg)  
  
cv\_linear\_1 <- cv.glm(wrigley\_agg, basic\_linear\_1, K = 10)  
summary(basic\_linear\_1)

##   
## Call:  
## glm(formula = Checkouts ~ Weekday + Game\_Soon + Hour + Temp +   
## Humidity + Rain\_Pouring + DOY, data = wrigley\_agg)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -126.61 -27.21 -5.89 15.25 291.29   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 15.160602 3.037977 4.990 6.16e-07 \*\*\*  
## WeekdayMONDAY -0.617149 1.811227 -0.341 0.73331   
## WeekdaySATURDAY 14.133462 1.786729 7.910 2.94e-15 \*\*\*  
## WeekdaySUNDAY 9.779627 1.816004 5.385 7.45e-08 \*\*\*  
## WeekdayTHURSDAY -4.283952 1.792556 -2.390 0.01688 \*   
## WeekdayTUESDAY -1.446152 1.784902 -0.810 0.41784   
## WeekdayWEDNESDAY -5.710803 1.778638 -3.211 0.00133 \*\*   
## Game\_Soon 46.476847 1.945068 23.895 < 2e-16 \*\*\*  
## Hour 1.917255 0.072359 26.497 < 2e-16 \*\*\*  
## Temp 2.160606 0.047300 45.679 < 2e-16 \*\*\*  
## Humidity -0.289604 0.034360 -8.428 < 2e-16 \*\*\*  
## Rain\_Pouring -26.983885 2.727370 -9.894 < 2e-16 \*\*\*  
## DOY 0.004241 0.004733 0.896 0.37023   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 1759.068)  
##   
## Null deviance: 22553267 on 7514 degrees of freedom  
## Residual deviance: 13196527 on 7502 degrees of freedom  
## AIC: 77498  
##   
## Number of Fisher Scoring iterations: 2

print(cv\_linear\_1$delta) #outputs the mean squared error from 10-fold cross validation

## [1] 1762.849 1762.490

While many of the coefficients have highly significant predictors, the model doesn't perform too well. This is not surprising given the exploratory data analysis, where we uncovered that the day of the year and the hour of the day affect the number of bikes checked out in a non-linear fashion. Using this, we will try to create a better linear model, transforming the variables.

basic\_linear\_2 <- glm(Checkouts ~ Game\_Soon + Weekend + poly(Hour, 6):Weekend + poly(Hour, 6) + Temp + Rain\_Pouring + poly(DOY, 5), data = wrigley\_agg)  
  
cv\_linear\_2 <- cv.glm(wrigley\_agg, basic\_linear\_2, K = 10)  
summary(basic\_linear\_2)

##   
## Call:  
## glm(formula = Checkouts ~ Game\_Soon + Weekend + poly(Hour, 6):Weekend +   
## poly(Hour, 6) + Temp + Rain\_Pouring + poly(DOY, 5), data = wrigley\_agg)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -96.241 -19.489 -3.984 14.870 291.866   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.961e+01 1.095e+00 27.026 < 2e-16 \*\*\*  
## Game\_Soon 3.086e+01 1.541e+00 20.030 < 2e-16 \*\*\*  
## Weekend 1.348e+01 8.327e-01 16.185 < 2e-16 \*\*\*  
## poly(Hour, 6)1 1.182e+03 3.880e+01 30.458 < 2e-16 \*\*\*  
## poly(Hour, 6)2 -1.255e+03 3.792e+01 -33.098 < 2e-16 \*\*\*  
## poly(Hour, 6)3 -4.846e+02 3.828e+01 -12.661 < 2e-16 \*\*\*  
## poly(Hour, 6)4 -5.629e+02 3.805e+01 -14.795 < 2e-16 \*\*\*  
## poly(Hour, 6)5 -7.592e+02 3.808e+01 -19.939 < 2e-16 \*\*\*  
## poly(Hour, 6)6 1.008e+03 3.792e+01 26.571 < 2e-16 \*\*\*  
## Temp 1.207e+00 7.757e-02 15.564 < 2e-16 \*\*\*  
## Rain\_Pouring -3.597e+01 2.049e+00 -17.554 < 2e-16 \*\*\*  
## poly(DOY, 5)1 2.773e+02 3.761e+01 7.373 1.85e-13 \*\*\*  
## poly(DOY, 5)2 -9.616e+02 6.804e+01 -14.132 < 2e-16 \*\*\*  
## poly(DOY, 5)3 -3.591e+02 3.752e+01 -9.570 < 2e-16 \*\*\*  
## poly(DOY, 5)4 2.035e+02 3.377e+01 6.027 1.75e-09 \*\*\*  
## poly(DOY, 5)5 -4.753e+01 3.259e+01 -1.459 0.145   
## Weekend:poly(Hour, 6)1 4.005e+02 7.291e+01 5.493 4.07e-08 \*\*\*  
## Weekend:poly(Hour, 6)2 -1.097e+03 7.286e+01 -15.058 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)3 -1.069e+03 7.249e+01 -14.746 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)4 1.727e+03 7.224e+01 23.904 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)5 1.264e+03 7.222e+01 17.507 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)6 -1.395e+03 7.255e+01 -19.224 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 1044.316)  
##   
## Null deviance: 22553267 on 7514 degrees of freedom  
## Residual deviance: 7825062 on 7493 degrees of freedom  
## AIC: 73588  
##   
## Number of Fisher Scoring iterations: 2

print(cv\_linear\_2$delta) #outputs the mean squared error from 10-fold cross validation

## [1] 1048.782 1048.385

Here, we get a substantially better fit from the modifications from the formula. Taking these factors into account lead to all significant predictors and a much lower MSE.

Now, we will try some models that may be more appripriate for count data. First, we will try a poisson regression model. We will log transform the Temperature since in this model the response is log transformed, so predictors that have a strong linear relationship with the response should also be log transformed.

basic\_poisson <- glm(Checkouts ~ Game\_Soon + Weekend + poly(Hour, 6):Weekend + poly(Hour, 6) + log(Temp + 1 - min(wrigley\_agg$Temp)) + Rain\_Pouring + poly(DOY, 5), data = wrigley\_agg, family = poisson)  
  
summary(basic\_poisson)

##   
## Call:  
## glm(formula = Checkouts ~ Game\_Soon + Weekend + poly(Hour, 6):Weekend +   
## poly(Hour, 6) + log(Temp + 1 - min(wrigley\_agg$Temp)) + Rain\_Pouring +   
## poly(DOY, 5), family = poisson, data = wrigley\_agg)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -11.981 -1.941 -0.456 1.296 48.944   
##   
## Coefficients:  
## Estimate Std. Error z value  
## (Intercept) 0.223704 0.038402 5.825  
## Game\_Soon 0.176384 0.004606 38.293  
## Weekend 0.280409 0.005974 46.941  
## poly(Hour, 6)1 65.230123 0.377729 172.690  
## poly(Hour, 6)2 -59.692799 0.312894 -190.777  
## poly(Hour, 6)3 1.010445 0.243859 4.144  
## poly(Hour, 6)4 4.466512 0.276860 16.133  
## poly(Hour, 6)5 -41.193838 0.280997 -146.599  
## poly(Hour, 6)6 42.385987 0.259470 163.356  
## log(Temp + 1 - min(wrigley\_agg$Temp)) 0.874537 0.011169 78.302  
## Rain\_Pouring -1.107920 0.015824 -70.015  
## poly(DOY, 5)1 6.556083 0.242651 27.019  
## poly(DOY, 5)2 -29.400476 0.362062 -81.203  
## poly(DOY, 5)3 -8.682520 0.229028 -37.910  
## poly(DOY, 5)4 -1.800249 0.187331 -9.610  
## poly(DOY, 5)5 -7.068344 0.183656 -38.487  
## Weekend:poly(Hour, 6)1 -10.882784 0.628404 -17.318  
## Weekend:poly(Hour, 6)2 8.871785 0.490788 18.077  
## Weekend:poly(Hour, 6)3 -32.737980 0.425084 -77.015  
## Weekend:poly(Hour, 6)4 29.357058 0.482857 60.799  
## Weekend:poly(Hour, 6)5 27.307667 0.468606 58.274  
## Weekend:poly(Hour, 6)6 -41.211686 0.431463 -95.516  
## Pr(>|z|)   
## (Intercept) 5.70e-09 \*\*\*  
## Game\_Soon < 2e-16 \*\*\*  
## Weekend < 2e-16 \*\*\*  
## poly(Hour, 6)1 < 2e-16 \*\*\*  
## poly(Hour, 6)2 < 2e-16 \*\*\*  
## poly(Hour, 6)3 3.42e-05 \*\*\*  
## poly(Hour, 6)4 < 2e-16 \*\*\*  
## poly(Hour, 6)5 < 2e-16 \*\*\*  
## poly(Hour, 6)6 < 2e-16 \*\*\*  
## log(Temp + 1 - min(wrigley\_agg$Temp)) < 2e-16 \*\*\*  
## Rain\_Pouring < 2e-16 \*\*\*  
## poly(DOY, 5)1 < 2e-16 \*\*\*  
## poly(DOY, 5)2 < 2e-16 \*\*\*  
## poly(DOY, 5)3 < 2e-16 \*\*\*  
## poly(DOY, 5)4 < 2e-16 \*\*\*  
## poly(DOY, 5)5 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)1 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)2 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)3 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)4 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)5 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)6 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 402286 on 7514 degrees of freedom  
## Residual deviance: 63706 on 7493 degrees of freedom  
## AIC: 100579  
##   
## Number of Fisher Scoring iterations: 5

cv\_poisson <- cv.glm(wrigley\_agg, basic\_poisson, K = 10)  
print(cv\_poisson$delta)

## [1] 577.3823 577.0869

Again, all of the coefficients in the model are significant, and we can see that this leads to a substantial improvement in the cross-validated mean squared error.

Now, we will try to regularize this model to correct for any overfitting on the higher order terms.

wrigley\_agg\_predictors <- model.matrix(Checkouts ~ Game\_Soon + Weekend + poly(Hour, 6):Weekend + poly(Hour, 6) + log(Temp + 1 - min(wrigley\_agg$Temp)) + Rain\_Pouring + poly(DOY, 5), data = wrigley\_agg)  
  
cv\_lasso\_pois <- cv.glmnet(wrigley\_agg\_predictors, wrigley\_agg$Checkouts, family = "poisson", alpha = 1, nlambda = 100, type.measure = "mse")  
  
mse\_min\_pois <- cv\_lasso\_pois$cvm[cv\_lasso\_pois$lambda == cv\_lasso\_pois$lambda.min]  
print(mse\_min\_pois)

## [1] 575.6786

best\_fit <- glmnet(wrigley\_agg\_predictors, wrigley\_agg$Checkouts, family = "poisson", alpha = 1, lambda = cv\_lasso\_pois$lambda.min)  
print(coef(best\_fit))

## 23 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 0.2547425  
## (Intercept) .   
## Game\_Soon 0.1753461  
## Weekend 0.2373138  
## poly(Hour, 6)1 61.8207776  
## poly(Hour, 6)2 -57.2883908  
## poly(Hour, 6)3 0.1271253  
## poly(Hour, 6)4 3.2724535  
## poly(Hour, 6)5 -39.3376834  
## poly(Hour, 6)6 40.5479492  
## log(Temp + 1 - min(wrigley\_agg$Temp)) 0.8748193  
## Rain\_Pouring -1.0849979  
## poly(DOY, 5)1 6.4383376  
## poly(DOY, 5)2 -28.9299035  
## poly(DOY, 5)3 -8.4330384  
## poly(DOY, 5)4 -1.5193363  
## poly(DOY, 5)5 -6.7430701  
## Weekend:poly(Hour, 6)1 -5.8159752  
## Weekend:poly(Hour, 6)2 5.1541471  
## Weekend:poly(Hour, 6)3 -31.5154191  
## Weekend:poly(Hour, 6)4 31.0329372  
## Weekend:poly(Hour, 6)5 24.4541911  
## Weekend:poly(Hour, 6)6 -38.4929404

This leads to virtually the same model with the same performance. The regularization does little. So, the poisson model led to significant improvements in the performance of the model. However, the data is heavily overdispersed, so a poisson model might not be best.

mean(wrigley\_agg$Checkouts)

## [1] 49.74584

var(wrigley\_agg$Checkouts)

## [1] 3001.499

To try to correct for this, a negative binomial regression model may be more appropriate, which allows the mean and the variance to be estimated separately.

neg\_binom\_model <- glm.nb(formula = Checkouts ~ Game\_Soon + Weekend + poly(Hour, 6):Weekend + poly(Hour, 6) + log(Temp + 1 - min(wrigley\_agg$Temp)) + Rain\_Pouring + poly(DOY, 5), data = wrigley\_agg)  
  
summary(neg\_binom\_model)

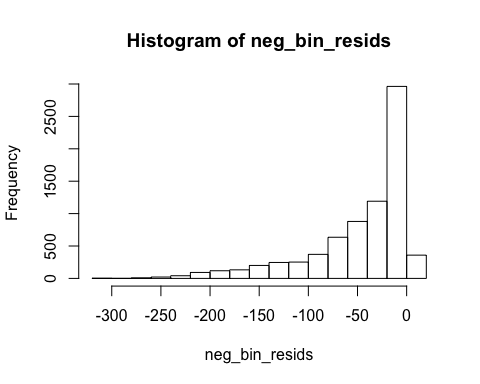
##   
## Call:  
## glm.nb(formula = Checkouts ~ Game\_Soon + Weekend + poly(Hour,   
## 6):Weekend + poly(Hour, 6) + log(Temp + 1 - min(wrigley\_agg$Temp)) +   
## Rain\_Pouring + poly(DOY, 5), data = wrigley\_agg, init.theta = 4.561166844,   
## link = log)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.2939 -0.8112 -0.1862 0.4336 24.7947   
##   
## Coefficients:  
## Estimate Std. Error z value  
## (Intercept) 0.60859 0.09439 6.447  
## Game\_Soon 0.24561 0.02306 10.652  
## Weekend 0.19368 0.01391 13.923  
## poly(Hour, 6)1 60.30027 0.73027 82.573  
## poly(Hour, 6)2 -56.83486 0.67451 -84.261  
## poly(Hour, 6)3 -3.40856 0.63043 -5.407  
## poly(Hour, 6)4 5.97544 0.64740 9.230  
## poly(Hour, 6)5 -41.34130 0.67294 -61.434  
## poly(Hour, 6)6 34.97867 0.67197 52.054  
## log(Temp + 1 - min(wrigley\_agg$Temp)) 0.77393 0.02774 27.901  
## Rain\_Pouring -1.06759 0.03641 -29.323  
## poly(DOY, 5)1 8.43330 0.61523 13.707  
## poly(DOY, 5)2 -30.39296 0.96597 -31.464  
## poly(DOY, 5)3 -11.17660 0.58808 -19.005  
## poly(DOY, 5)4 -1.81085 0.55004 -3.292  
## poly(DOY, 5)5 -5.29352 0.54487 -9.715  
## Weekend:poly(Hour, 6)1 -9.54703 1.28343 -7.439  
## Weekend:poly(Hour, 6)2 9.07112 1.21164 7.487  
## Weekend:poly(Hour, 6)3 -31.65813 1.18364 -26.746  
## Weekend:poly(Hour, 6)4 30.35687 1.20211 25.253  
## Weekend:poly(Hour, 6)5 31.42089 1.21469 25.867  
## Weekend:poly(Hour, 6)6 -41.04917 1.21882 -33.679  
## Pr(>|z|)   
## (Intercept) 1.14e-10 \*\*\*  
## Game\_Soon < 2e-16 \*\*\*  
## Weekend < 2e-16 \*\*\*  
## poly(Hour, 6)1 < 2e-16 \*\*\*  
## poly(Hour, 6)2 < 2e-16 \*\*\*  
## poly(Hour, 6)3 6.42e-08 \*\*\*  
## poly(Hour, 6)4 < 2e-16 \*\*\*  
## poly(Hour, 6)5 < 2e-16 \*\*\*  
## poly(Hour, 6)6 < 2e-16 \*\*\*  
## log(Temp + 1 - min(wrigley\_agg$Temp)) < 2e-16 \*\*\*  
## Rain\_Pouring < 2e-16 \*\*\*  
## poly(DOY, 5)1 < 2e-16 \*\*\*  
## poly(DOY, 5)2 < 2e-16 \*\*\*  
## poly(DOY, 5)3 < 2e-16 \*\*\*  
## poly(DOY, 5)4 0.000994 \*\*\*  
## poly(DOY, 5)5 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)1 1.02e-13 \*\*\*  
## Weekend:poly(Hour, 6)2 7.06e-14 \*\*\*  
## Weekend:poly(Hour, 6)3 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)4 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)5 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)6 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for Negative Binomial(4.5612) family taken to be 1)  
##   
## Null deviance: 44534.8 on 7514 degrees of freedom  
## Residual deviance: 8645.8 on 7493 degrees of freedom  
## AIC: 60380  
##   
## Number of Fisher Scoring iterations: 1  
##   
##   
## Theta: 4.5612   
## Std. Err.: 0.0966   
##   
## 2 x log-likelihood: -60334.3900

cv\_nb <- cv.glm(wrigley\_agg, neg\_binom\_model, K = 10)  
print(cv\_nb$delta)

## [1] 647.7347 647.2676

However, for some reason this does not actually improve the fit of the regression model at all. This may be because of the excess of 0 counts in the data. Here, we can see that the residuals are very negative for the data.

neg\_bin\_resids <- predict(neg\_binom\_model, data = wrigley\_agg) - wrigley\_agg$Checkouts  
hist(neg\_bin\_resids)



mean(neg\_bin\_resids)

## [1] -46.49851

To correct for this, we will try to use a zero-inflated model that takes into account an excess zero-count in the data. We use a hurdle model, which in this package uses a point mass at zero in addition to the poisson distribution.

bike\_zeroinfl\_wrig <- hurdle(formula = Checkouts ~ Game\_Soon + Weekend + poly(Hour, 6):Weekend + poly(Hour, 6) + log(Temp + 1 - min(wrigley\_agg$Temp)) + Rain\_Pouring + poly(DOY, 5), data = wrigley\_agg, dist = "poisson")  
  
summary(bike\_zeroinfl\_wrig)

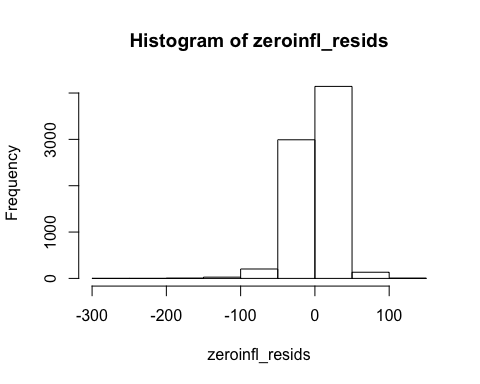
##   
## Call:  
## hurdle(formula = Checkouts ~ Game\_Soon + Weekend + poly(Hour, 6):Weekend +   
## poly(Hour, 6) + log(Temp + 1 - min(wrigley\_agg$Temp)) + Rain\_Pouring +   
## poly(DOY, 5), data = wrigley\_agg, dist = "poisson")  
##   
## Pearson residuals:  
## Min 1Q Median 3Q Max   
## -9.5433 -1.6028 -0.4122 1.3462 234.9620   
##   
## Count model coefficients (truncated poisson with log link):  
## Estimate Std. Error z value  
## (Intercept) 0.231552 0.038697 5.984  
## Game\_Soon 0.176631 0.004607 38.341  
## Weekend 0.275234 0.006117 44.993  
## poly(Hour, 6)1 64.399051 0.396797 162.297  
## poly(Hour, 6)2 -59.110742 0.324056 -182.409  
## poly(Hour, 6)3 1.065959 0.245874 4.335  
## poly(Hour, 6)4 4.004336 0.283435 14.128  
## poly(Hour, 6)5 -40.571585 0.291405 -139.227  
## poly(Hour, 6)6 41.966386 0.268319 156.405  
## log(Temp + 1 - min(wrigley\_agg$Temp)) 0.874834 0.011245 77.800  
## Rain\_Pouring -1.100987 0.015877 -69.344  
## poly(DOY, 5)1 6.640947 0.243539 27.269  
## poly(DOY, 5)2 -29.298964 0.363623 -80.575  
## poly(DOY, 5)3 -8.577051 0.230317 -37.240  
## poly(DOY, 5)4 -1.779314 0.187965 -9.466  
## poly(DOY, 5)5 -7.019329 0.184225 -38.102  
## Weekend:poly(Hour, 6)1 -10.260412 0.644446 -15.921  
## Weekend:poly(Hour, 6)2 8.450248 0.499402 16.921  
## Weekend:poly(Hour, 6)3 -32.686539 0.426967 -76.555  
## Weekend:poly(Hour, 6)4 29.737476 0.489598 60.739  
## Weekend:poly(Hour, 6)5 26.732337 0.477295 56.008  
## Weekend:poly(Hour, 6)6 -40.726444 0.437831 -93.019  
## Pr(>|z|)   
## (Intercept) 2.18e-09 \*\*\*  
## Game\_Soon < 2e-16 \*\*\*  
## Weekend < 2e-16 \*\*\*  
## poly(Hour, 6)1 < 2e-16 \*\*\*  
## poly(Hour, 6)2 < 2e-16 \*\*\*  
## poly(Hour, 6)3 1.46e-05 \*\*\*  
## poly(Hour, 6)4 < 2e-16 \*\*\*  
## poly(Hour, 6)5 < 2e-16 \*\*\*  
## poly(Hour, 6)6 < 2e-16 \*\*\*  
## log(Temp + 1 - min(wrigley\_agg$Temp)) < 2e-16 \*\*\*  
## Rain\_Pouring < 2e-16 \*\*\*  
## poly(DOY, 5)1 < 2e-16 \*\*\*  
## poly(DOY, 5)2 < 2e-16 \*\*\*  
## poly(DOY, 5)3 < 2e-16 \*\*\*  
## poly(DOY, 5)4 < 2e-16 \*\*\*  
## poly(DOY, 5)5 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)1 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)2 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)3 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)4 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)5 < 2e-16 \*\*\*  
## Weekend:poly(Hour, 6)6 < 2e-16 \*\*\*  
## Zero hurdle model coefficients (binomial with logit link):  
## Estimate Std. Error z value  
## (Intercept) 5.71287 1.22626 4.659  
## Game\_Soon 11.08884 583.73834 0.019  
## Weekend -1.06207 1.58340 -0.671  
## poly(Hour, 6)1 120.98190 27.67853 4.371  
## poly(Hour, 6)2 -362.81780 95.73279 -3.790  
## poly(Hour, 6)3 43.00507 36.78579 1.169  
## poly(Hour, 6)4 225.78922 61.87537 3.649  
## poly(Hour, 6)5 -132.24945 45.17835 -2.927  
## poly(Hour, 6)6 -101.66814 31.50410 -3.227  
## log(Temp + 1 - min(wrigley\_agg$Temp)) 1.06138 0.19886 5.337  
## Rain\_Pouring -1.42227 0.35050 -4.058  
## poly(DOY, 5)1 -0.08844 6.24894 -0.014  
## poly(DOY, 5)2 -36.48480 9.37313 -3.892  
## poly(DOY, 5)3 -22.87580 6.41754 -3.565  
## poly(DOY, 5)4 11.56415 6.24277 1.852  
## poly(DOY, 5)5 1.61859 6.17209 0.262  
## Weekend:poly(Hour, 6)1 -32.16236 36.05686 -0.892  
## Weekend:poly(Hour, 6)2 65.78156 131.28952 0.501  
## Weekend:poly(Hour, 6)3 -130.73948 51.99620 -2.514  
## Weekend:poly(Hour, 6)4 -17.29814 84.90500 -0.204  
## Weekend:poly(Hour, 6)5 174.79914 55.53065 3.148  
## Weekend:poly(Hour, 6)6 -14.72192 46.65440 -0.316  
## Pr(>|z|)   
## (Intercept) 3.18e-06 \*\*\*  
## Game\_Soon 0.984844   
## Weekend 0.502377   
## poly(Hour, 6)1 1.24e-05 \*\*\*  
## poly(Hour, 6)2 0.000151 \*\*\*  
## poly(Hour, 6)3 0.242376   
## poly(Hour, 6)4 0.000263 \*\*\*  
## poly(Hour, 6)5 0.003419 \*\*   
## poly(Hour, 6)6 0.001250 \*\*   
## log(Temp + 1 - min(wrigley\_agg$Temp)) 9.43e-08 \*\*\*  
## Rain\_Pouring 4.95e-05 \*\*\*  
## poly(DOY, 5)1 0.988709   
## poly(DOY, 5)2 9.92e-05 \*\*\*  
## poly(DOY, 5)3 0.000364 \*\*\*  
## poly(DOY, 5)4 0.063967 .   
## poly(DOY, 5)5 0.793134   
## Weekend:poly(Hour, 6)1 0.372398   
## Weekend:poly(Hour, 6)2 0.616342   
## Weekend:poly(Hour, 6)3 0.011923 \*   
## Weekend:poly(Hour, 6)4 0.838560   
## Weekend:poly(Hour, 6)5 0.001645 \*\*   
## Weekend:poly(Hour, 6)6 0.752342   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1   
##   
## Number of iterations in BFGS optimization: 28   
## Log-likelihood: -5.006e+04 on 44 Df

cv\_zero <- cv.glm(wrigley\_agg, bike\_zeroinfl\_wrig, K = 10)  
print(cv\_zero$delta)

## [1] 575.3592 575.1283

This leads leads to essentially the same mean squared error as the non zero-inflated model. However, it provides a much more useful model. In the previous model, all of the estimates were extremely low, where now the residuals are centered. It is better to have an overestimate instead of an underestimate in the case of Divvy bikes. If Divvy is unprepared for a massive spike in demand, this could lead to a large decrease in business. If, however, they are a bit overprepared in some cases, then they aren't hurt in any significant way.

zeroinfl\_resids <- predict(bike\_zeroinfl\_wrig, data = wrigley\_agg) - wrigley\_agg$Checkouts  
hist(zeroinfl\_resids)



mean(zeroinfl\_resids)

## [1] -0.02643696

Now, we will try some more advanced machine learning methods. First, we will try to train a random forest. We use the default 500 trees, and at each step we randomly select five sevenths of the original variables. Now, we will change the variables that we are training on as well. Instead of having a variable represent if there is a cubs game within an 8 hour radius, we will simply input the time of the game and let the algorithm undrestand the relation of the time of the cubs game with the other variables itself. We will also put rain intensity back in the model, because despite the fact that there are strange outliers, random forests are affected much less by these outliers.

train\_wrigley\_agg <- data.frame(wrigley\_agg$Temp, wrigley\_agg$Hour, wrigley\_agg$DOY, wrigley\_agg$Weekday, wrigley\_agg$Rain\_Intensity, wrigley\_agg$Cubs\_Game, wrigley\_agg$Humidity)  
  
# this function defines the proportion of variables that should be used in the random forests  
p\_red <- function(p){  
   
 p\_red <- p\*(4/7)  
 next\_param <- ceiling(p\_red)  
 return(next\_param)  
   
}  
  
rfcv\_test2 <- rfcv(train\_wrigley\_agg, wrigley\_agg$Checkouts, mtry = p\_red, cv.fold = 10)  
print(rfcv\_test2$error.cv)

## 7 4 1   
## 274.4181 339.4539 2044.2041

The first element of the output represents training a random forest with all 7 of the original variables. This leads to a very substantial decrease in the cross-validated mean squared error, decreasing it by more than 50%.

Summary

The random forest is the best method to predict the number of bikes that are rented every hour.

Below is the final model. The importance function measures how much adding different dimensions will increase prediction accuracy. We can see that the hour of the day is the most important predictor, which is not surprising because bike riding, regardless of other factors, is heavily affected by the hour of the day. Very few bikes are ridden at 4AM. Beyond this, we can see that the weather is also an important factor, as predicted. Finally, we can see that the time of the Cubs game also has a substantial effect of the number of bikes checked out. While it is not the most important factor, the predictions are made substantially better by the inclusion of this variable.

rf\_predictive <- randomForest(formula = Checkouts ~ Temp + Hour + DOY + Weekday + Rain\_Intensity + Cubs\_Game + Humidity, data = wrigley\_agg, mtry = 4)  
  
importance(rf\_predictive)

## IncNodePurity  
## Temp 5989136.5  
## Hour 10728585.7  
## DOY 2147339.5  
## Weekday 1704674.7  
## Rain\_Intensity 209158.1  
## Cubs\_Game 433877.6  
## Humidity 985280.1

Finally, we can estimate the mean absolute error of the model using a validation set. Unfortunately, there is no option to evaluate the mean absolute error in the rfcv function. However, this provides a good estimate. On average, this model is only about 10 bikes off of the true number of bikes that are rented. A better model could definitely be constructed using other, more advanced methods, but this model would give Divvy bike substantial predictive power.

sample <- sample.int(n = nrow(wrigley\_agg), size = floor(.75\*nrow(wrigley\_agg)), replace = F)  
train\_data <- wrigley\_agg[sample, ]  
test\_data <- wrigley\_agg[-sample, ]  
rf\_cv\_test <- randomForest(formula = Checkouts ~ Temp + Hour + DOY + Weekday + Rain\_Intensity + Cubs\_Game + Humidity, data = train\_data, mtry = 4)  
preds\_rf <- predict(rf\_cv\_test, newdata = test\_data)  
resids <- preds\_rf - test\_data$Checkouts  
abs\_resids <- abs(resids)  
mean(abs\_resids)

## [1] 10.61486

Conclusion: What is a random forest, and why did it work here?

Unfortunately, no rigorous reason can be given for why the random forest performed better than the other machine learning methods that were tried in the model. We only have the test performance to judge the validity of the model. However, looking back to the exploratory data analysis, some insights can be made. In order to do this, though, it is first important to explain the concept of a random forest for those who are not already familiar.

The basis of random forests are tree based methods. In this case, recursive regression trees. A regression tree is a highly flexible learning method. The algorithm successively divides up the feature space into a number of different horizontal planes/regions, and then for each region it predicts any point in that region to be the mean of all points in the region. The tree is recursive, because instead of making a globally optimal choice about how to split up the feature space, the algorithm makes the locally optimal choice at each step, splitting the region that will minimize the loss function at each step.

Because this method is so flexible, it often suffers from high variance. So, instead of just using a tree, I decided to try a random forest. A random forest is an ensemble method for training trees. The point of random forests is to decrease the variance of the trees, in order to reduce the mean squared error. This method is a slight modification to bagging. Bagging, short for bootstrap aggregation, is a powerful method of variance reduction. When using a bagged model, one trains a number of different models, where each model is trained on a bootstrapped portion of the original data set instead of the entire set (bootstrapped just means randomly drawn with replacement). Then, after all of the models are trained, their predictions are aggregated and averaged. In theory, this method would substantially reduce variance because of the high number of models that are combined together. However, in practice this variance reduction is weakened because many of the models are highly correlated, even though the training data for each model is different because of the bootstrapping. If virtually the same model is created and averaged 100 times, then this won't actually lead to any difference in prediction. To avoid this issue, a random forest decorrelates the trees used in the boostrap aggregation. To decorrelate the trees, a random forest forces the trees to make different splits. At each step, instead of the algorithm selecting from all of the dimensions to decide what split in the sample space to make, the algorithm only selects from a randomly selected portion of the dimensions. While this will slightly increase the bias of the trees, it can substantially reduce the overall variance in the aggregated model because the trees are less correlated.

Because many of the predictors in this dataset did not have a simple linear relationship with the response variable, a random forest performed much better. Adding complex polynomial and interactions terms substantially increased the performance of the regression models. However, this can only be done so far until overfitting occurs. Because the random forest is so flexible, it was able to simply capture the relationships encoded in the complex polynomial and interaction terms, and it was able to explain even more of the variance in the data than the regression models. This is the power of random forests.