Forecasting Bike-Share Usage via Conditional Inference Trees and Random Forests

Andrew Trice

April 29, 2015

1 Introduction

A bike-share is a system in which bike are rented on a on-demand basis. They can be returned when and where they are no longer needed. They are paid for through online kiosks that record arrival location, trip length, trip duration, and departure location. This data is important because it can give researchers an indication of the transportation patterns of a city. These transportation patterns can be extrapolated to the city's vitality, naunces, and improvements that are needed

2 Materials

This is a purely research based competition hosted on the competitive Data Science platform, Kaggle. The dataset was provided by Hadi Fanaee Tork using data from Capital One Bikeshare and is hosted by the UCI machine learning library.

Capital One Bikeshare in Washington DC presents bikeshare rental data from January 1st 2011 to December 31st 2012. The season, whether it was a holiday, whether it was a workingday, severity of the weather, temperature, adjusted temperature, humidity, windspeed, casual rentals, and registered rentals are all included as features. The goal of the competition is to predict bicycle rental count for each individual hour.

We are using the R open-source statistical programming language and Rstudio software to conduct analysis. The data is already partitioned into training and testing sets in csv or comma separated value format. We import the "party" and "randomForests" packages from R.

3 Methods

3.1 Feature Engineering

All data given was numerical and different regression methods could suffice to forecast bike rentals. But we want to apply novel methods of forecasting demand by first using a decision tree to establish as a benchmark and then by creating a random forests ensemble.

First though we must feature engineer our data into more useful variables. The datetime feature contains the date in y-m-d form followed by the time. We substring the time from datetime and then an integer variable hour for the particular hour, we convert hour back to a factor. We use the weekdays and as.date functions on the datetime variable to create a day of the week factor variable. We factor the weather, season, and working day variables. We notice that there are minimal instances of weather = 4 so we reassign those variables to weather = 3 to increase model robustness.

With these newly engineered variables we can view average rental times by hour and day of the week and understand their effect on bike rentals.

3.2 Initial Plot Analysis

We notice that Saturday and Sunday have significantly different times than the weekdays. The weekdays have the most pronounced rental periods at 8AM and 5PM where the weekend has a gradual period at 10AM through 5PM. We proceed to build our our initial decision tree.

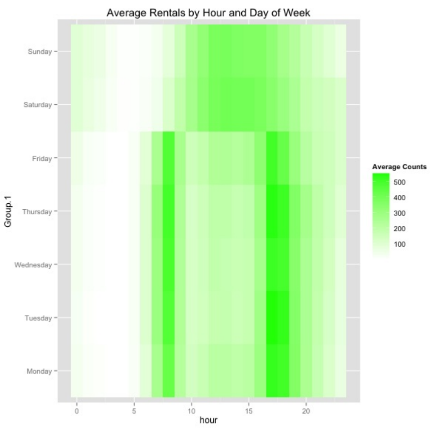

Figure 1: Bike Rental Averages by Day of Week and Hour

3.3 Conditional Inference Trees

3.3.1 Decision Tree Overview

Decision trees are like flowcharts in which each node represents the class label and each branch represents the decision made. Extrapolated to our case we may see a decision tree node of weather. If the weather is 3 or 4 then we will see a reduction in the number of bikes rented. A tree may have multiple nodes and branches to make a conclusion.

Decision trees are commonly used in operations research to determine optimal strategy but have applications to many fields. They are especially useful for determining variable importance in our forecasting methods.

3.3.2 Conditional Inference Tree Overview

We choose to implement a conditional inference tree, or ctree, as a benchmark over the traditional inference tree. The ctree avoids some of the variable selection bias of the traditional decision tree by using a significance test procedure to select variables as opposed to selecting the variables that maximize information gain.

3.3.3 Conditional Inference Tree Implementation

Our ctree is built with the variables: season, holiday, workingday, weather, temp, atemp, humidity, and hour. This builds an extensive tree with 393 individual nodes.

3.4 Random Forests

3.4.1 Random Forests Overview

Random forests construct many decision trees and output the mean prediction of the decision trees it builds. Each decision tree is constructed optimally from a random subset of the features in what is called "feature bagging" or "bagging". This helps correct for overfitting or when a model simply memorizes the training data as opposed to detecting the dependent variable relationships.

$$\hat{f} = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x')$$

Figure 2: Random Forests Unseen Predictions

$$\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^{n}$$

Figure 3: Random Forests Variable Importance

3.4.2 Random Forests Implementation

We use the same variables as before except we are going to create two formulas: one for the casual rentals and one for the registered rentals. We hypothesize that the random forests can pick up on the nuances of the two rental types. We set the seed to be 192 or the mean rental of the training data, number of trees in the Forests to be 2000 each, the number of variables per level to be 2, which is the default, and for importance = true meaning high variable importance.

4 Results

4.1 Conditional Inference Tree

The submission of the conditional inference tree received a rank of 950 out of 2219 distinct teams at the time of submission. This makes it better than 57% of teams. The average bike rental value is 189.9 which differs only marginally from the training data which is 191.6. This is an appropriate benchmark and sets a reasonable standard for the random forest implementation.

4.2 Random Forests

At the time of this publication the random forests receives a placement of 479 out of 2778. This makes it better than 82% of teams. We can see that our random forests receives a significantly improved score over the conditional inference tree benchmark and we can observe the variables' importance.

4.3 Variable Importance

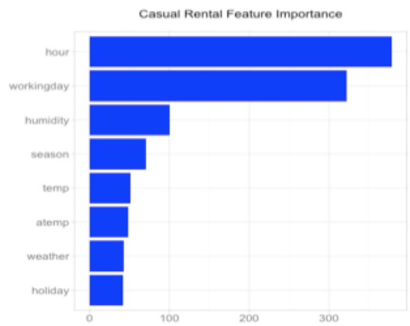

Figure 4: Casual Rental Variable Importance

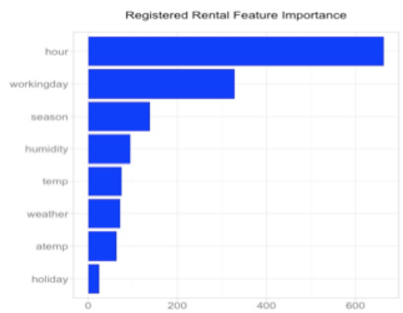

Figure 5: Registered Rental Variable Importance

Notice that in order of importance the variables for casual are: hour, workingday, humidity, season, temp, atemp, weather, and holiday. For registered the variables are: hour, workingday, season, humidity, temp, weather, atemp, and holiday.

The registered has a far greater importance on hour than the casual riders which primarily comes at the expense of the workingday feature.

5 References

Côme Etienne and Oukhellou Latifa. 2014. Model-Based Count Series Clustering for Bike Sharing System Usage Mining: A Case Study with the Vélib' System of Paris. ACM Trans. Intell. Syst. Technol. 5, 3, Article 39 (July 2014), 21 pages. DOI=10.1145/2560188 http://doi.acm.org/10.1145/2560188

Fanaee-T, Hadi, and Gama, Joao, Event labeling combining ensemble detectors and background knowledge, Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg. Rixey, R. Alexander. "Station-Level Fore-

casting of Bikesharing Ridership."Transportation Research Record: Journal of the Transportation Research Board2387.1 (2013): 46-55.

6 Appendix

6.1 Acknowledgments

- Brandon Harris for his excellent blog post on feature engineering and developing a basic Conditional Inference Tree.
- Kaggle user bruschkov for his forum post on a random forests tutorial.
- Beata Strubel for her ggplot rental times imaging.
- And the entire Kaggle community for the competition and knowledge fostering environment.

6.2 Conditional Inference Tree R Code

```
#package party is used to implement conditional inference tree
#package ggplot2 is used for visualizations
library('party')
library('ggplot2')

#set working directory
setwd("/users/andrewtrice/Desktop/Data Science Capstone")

#read in train/test
train <- read.csv("train.csv")
test <- read.csv("test.csv")

str(train)

#factorize training set
train_factor <- train
train_factor$weather <- factor(train$weather)
train_factor$holiday <- factor(train$holiday)
```

```r
train_factor$workingday <- factor(train$workingday)
train_factor$season <- factor(train$season)

#factorize test set
test_factor <- test
test_factor$weather <- factor(test$weather)
test_factor$holiday <- factor(test$holiday)
test_factor$workingday <- factor(test$workingday)
test_factor$season <- factor(test$season)

#create time column by stripping out timestamp
train_factor$time <- substring(train$datetime,12,20)
test_factor$time <- substring(test$datetime,12,20)

#factorize new timestamp column
train_factor$time <- factor(train_factor$time)
test_factor$time <- factor(test_factor$time)

#create day of week column
train_factor$day <- weekdays(as.Date(train_factor$datetime))
train_factor$day <- as.factor(train_factor$day)
test_factor$day <- weekdays(as.Date(test_factor$datetime))
test_factor$day <- as.factor(test_factor$day)

aggregate(train_factor[,"count"],list(train_factor$day),mean)

#create Sunday variable
train_factor$sunday[train_factor$day == "Sunday"] <- "1"
train_factor$sunday[train_factor$day != "1"] <- "0"

test_factor$sunday[test_factor$day == "Sunday"] <- "1"
test_factor$sunday[test_factor$day != "1"] <- "0"

#convert to factor
train_factor$sunday <- as.factor(train_factor$sunday)
test_factor$sunday <- as.factor(test_factor$sunday)

#convert time and create $hour as integer to evaluate for daypart
train_factor$hour<- as.numeric(substr(train_factor$time,1,2))
test_factor$hour<- as.numeric(substr(test_factor$time,1,2))

#create daypart column, default to 4 to make things easier for ourselves
train_factor$daypart <- "4"
test_factor$daypart <- "4"

#4AM - 10AM = 1
train_factor$daypart[(train_factor$hour < 10) & (train_factor$hour > 3)] <- 1
test_factor$daypart[(test_factor$hour < 10) & (test_factor$hour > 3)] <- 1

#11AM - 3PM = 2
train_factor$daypart[(train_factor$hour < 16) & (train_factor$hour > 9)] <- 2
test_factor$daypart[(test_factor$hour < 16) & (test_factor$hour > 9)] <- 2

#4PM - 9PM = 3
train_factor$daypart[(train_factor$hour < 22) & (train_factor$hour > 15)] <- 3
```

```r
test_factor$daypart[(test_factor$hour < 22) & (test_factor$hour >
    15)] <- 3

#convert daypart to factor
train_factor$daypart <- as.factor(train_factor$daypart)
test_factor$daypart <- as.factor(test_factor$daypart)

#convert hour back to factor
train_factor$hour <- as.factor(train_factor$hour)
test_factor$hour <- as.factor(test_factor$hour)

#create formulas for casual and registered riders
formula <- count ~ season + holiday + workingday + weather + temp +
     atemp + humidity + hour + daypart + sunday

#conditional inference tree
fit.ctree <- ctree(formula, data=train_factor)

#examine model for variable importance
fit.ctree

#run model against test data set
predict.ctree <- predict(fit.ctree, test_factor)

#label importance
imp <- importance(fit.ctree, type=1)
featureImportance <- data.frame(Feature=row.names(imp), Importance=
    imp[,1])

#plot variable importance
p <- ggplot(featureImportance, aes(x=reorder(Feature, Importance),
    y=Importance)) +
  geom_bar(stat="identity", fill="#12cfff") +
  coord_flip() +
  theme_light(base_size=20) +
  xlab("Importance") +
  ylab("") +
  ggtitle("Random Forest Feature Importance\n") +
  theme(plot.title=element_text(size=18))

#build a dataframe with our results
submit.ctree <- data.frame(datetime = test$datetime, count=ctree.
    count)

#write results to .csv for submission
write.table(submit.ctree, file="submit_ctree_v2.csv", sep=",",row.
    names=FALSE, col.names=v)
```

6.3 Random Forests R Code

```r
#import necessary packages
library('randomForest')
library('ggplot2')

#set working directory
setwd("/users/andrewtrice/Desktop/Data Science Capstone")

#import datasets from working directory
train <- read.csv("train.csv") #use nrows=1000 rows for speed
    during feature engineering
test <- read.csv("test.csv") #use nrows=1000 rows for speed during
    feature engineering
```

```r

str(train)

#factorize training set
train_factor <- train
train_factor$weather <- factor(train$weather)
train_factor$holiday <- factor(train$holiday)
train_factor$workingday <- factor(train$workingday)
train_factor$season <- factor(train$season)

#factorize test set
test_factor <- test
test_factor$weather <- factor(test$weather)
test_factor$holiday <- factor(test$holiday)
test_factor$workingday <- factor(test$workingday)
test_factor$season <- factor(test$season)

#create time column by stripping out timestamp
train_factor$time <- substring(train$datetime,12,20)
test_factor$time <- substring(test$datetime,12,20)

#factorize new timestamp column
train_factor$time <- factor(train_factor$time)
test_factor$time <- factor(test_factor$time)

#create day of week column
train_factor$day <- weekdays(as.Date(train_factor$datetime))
train_factor$day <- as.factor(train_factor$day)
test_factor$day <- weekdays(as.Date(test_factor$datetime))
test_factor$day <- as.factor(test_factor$day)

aggregate(train_factor[,"count"],list(train_factor$day),mean)

#create Sunday variable
train_factor$sunday[train_factor$day == "Sunday"] <- "1"
train_factor$sunday[train_factor$day != "1"] <- "0"

test_factor$sunday[test_factor$day == "Sunday"] <- "1"
test_factor$sunday[test_factor$day != "1"] <- "0"

#convert to factor
train_factor$sunday <- as.factor(train_factor$sunday)
test_factor$sunday <- as.factor(test_factor$sunday)

#convert time and create $hour as integer to evaluate for daypart
train_factor$hour<- as.numeric(substr(train_factor$time,1,2))
test_factor$hour<- as.numeric(substr(test_factor$time,1,2))

#create daypart column, default to 4 to make things easier for ourselves
train_factor$daypart <- "4"
test_factor$daypart <- "4"

#4AM - 10AM = 1
train_factor$daypart[(train_factor$hour < 10) & (train_factor$hour > 3)] <- 1
test_factor$daypart[(test_factor$hour < 10) & (test_factor$hour > 3)] <- 1

#11AM - 3PM = 2
train_factor$daypart[(train_factor$hour < 16) & (train_factor$hour > 9)] <- 2
```

```
test_factor$daypart[(test_factor$hour < 16) & (test_factor$hour > 9)] <- 2

#4PM - 9PM = 3
train_factor$daypart[(train_factor$hour < 22) & (train_factor$hour > 15)] <- 3
test_factor$daypart[(test_factor$hour < 22) & (test_factor$hour > 15)] <- 3

#convert daypart to factor
train_factor$daypart <- as.factor(train_factor$daypart)
test_factor$daypart <- as.factor(test_factor$daypart)

#convert hour back to factor
train_factor$hour <- as.factor(train_factor$hour)
test_factor$hour <- as.factor(test_factor$hour)

#get rid of weather 4
train$weather[train$weather==4] <- 3
test$weather[test$weather==4] <- 3

#variables
myNtree = 500
myMtry = 7
myImportance = TRUE

#set the seed to mean for model reproducibility
set.seed(192)

#fit and predict casual
casualFit <- randomForest(casual ~ season + holiday + workingday + weather + temp + atemp + humidity + hour, data=train_factor, ntree=myNtree, mtry=myMtry, importance=myImportance)
predictCasual <- predict(casualFit, test_factor)

#fit and predict registered
registeredFit <- randomForest(registered ~ season + holiday + workingday + weather + temp + atemp + humidity + hour, data=train_factor, ntree=myNtree, mtry=myMtry, importance=myImportance)
predictRegistered <- predict(registeredFit, test_factor)

#add both columns into final count, round to whole number
test$count <- casualFit + test$registered

#testplot
plot(train$count)
plot(test$count)

#write output to csv for submission
submit <- data.frame (datetime = test$datetime, count = test$count)
write.csv(submit, file = "randomForest_Prediction.csv", row.names=FALSE)

#label variable importances for both casual and registered
imp1 <- importance(casualFit, type=1)
imp2 <- importance(registeredFit, type=1)
featureImportance1 <- data.frame(Feature=row.names(imp1), Importance=imp1[,1])
featureImportance2 <- data.frame(Feature=row.names(imp2), Importance=imp2[,1])

```

```
#plot variable importances
casualImportance <- ggplot(featureImportance1, aes(x=reorder(
    Feature, Importance), y=Importance)) +
  geom_bar(stat="identity", fill="blue") +
  coord_flip() +
  theme_light(base_size=20) +
  xlab("") +
  ylab("") +
  ggtitle("Casual Rental Feature Importance\n") +
  theme(plot.title=element_text(size=18))

registeredImportance <- ggplot(featureImportance2, aes(x=reorder(
    Feature, Importance), y=Importance)) +
  geom_bar(stat="identity", fill="blue") +
  coord_flip() +
  theme_light(base_size=20) +
  xlab("") +
  ylab("") +
  ggtitle("Registered Rental Feature Importance\n") +
  theme(plot.title=element_text(size=18))
```