Homework 3 - BUSN 41204

Fernando Garcia, Paula Gaviria, Victor Fuentes

02/06/2021

Bike Sharing Demand - Kaggle Competence

In Section 1, we loaded the data and transformed the `count` variable into `log_count` to a better assessment of bike sharing demand usually behaved as an skewed distribution. Later, we started by examining the data graphically in section 2. We added a color aesthetic by season to the scatter plots between `count` and `windspeed`, `humidity`, `atemp`, and `temp`. Then, we noted outliers in the first three variables (Section 2.1.a). In particular, we observed twice `windspeed` repeated with the value of 56.9969. Moreover, we also realized that during Winter there was one day, March 10, with humidity equal to zero, which is not possible. Additionally, we found an outlier in `atemp`, a very low temperature in the Summer. We further explored these outliers to identify the exact date and time and replaced the outliers. For `windspeed`, we interpolated with the closest numbers, while for for `atemp` and `humidity`, we used an average between values by hours from the day before and the day after.

We continued exploring dataset by creating boxplots for both `season` and `weather` (Section 2.1.b). There, we found that in `weather` there were some half values, 1.5 and 2.5, which did not fall into any specific category. Using a conservative approach, we changed these values by rounding up to the nearest number with decimals so they can be representative of one of the four weather categories. Later on, we also evaluated the distribution of `log_count` by hour and `workingday`. There, we found a differentiate pattern by type of `workingday`. During `workingday = 1`, the peak is centered at 8am and 5pm-6pm. ON the other hand, on `workingday = 0` the peak is concentrated between 10am and 4pm. Additionally, we explored the distribution of count by hour and season finding a clear difference between Winter season and the other three ones. Finally, we also analyzed the evolution over time (`year` and `month`) highlighting that bike sharing demand should be growing or expanding between 2012 and 2011.

Formulating simplest RF and Boosting models and exploring their Variable Importance plots we found the grat importance of `hour` as the main predictor of `log_count`. While RF includes `workingday` as the second best predictor, GBM ranks it as the seventh one. Later, both models ranks `daylabel`, `atemp`, and `temp` in the same order. It's possible to hypothesize that differences in variable ranking could be based on interactions between explanatory variables (i.e., `workingday` and `hour`), high correlations between explanatory variables (i.e., `temp` and `atemp`), and/or the presence of seasonality in data (a weekly pattern not controlled yet).

Before developing ML models, we solved some previous data issues. After several trial and error attempts, we found that some days could be better identify as what people perceive in reality over what federal calendar indicates. In particular, we noticed an interesting pattern about Tax Day. When it's originally assigned on Fridays, it's moved to the next Monday. In this case, we appreciated that April 15th 2011 (Friday) and April 16th in 2012 (Friday) show a pattern like a regular `workingday`, while Monday 18th and Monday 19th show a pattern similar to a `holiday`. Hence, we recoded those days as `workingday = 1` and `holiday = 0`. Under this approach, we also recoded Friday after Thanksgiving, the Christmas Eve and the New Year Eve. We changed `workingday = 0` and `holiday = 1` in the `test` set.

After recoding the variables in our `train` and `test` data set we further explored the interaction and importance of our variables. We did this by constructing Partial Dependence Plots using a Random Forest model, a Boosting mode, individual Tree Regressions (`log_count ~ var_i`), and using a correlation matrix. So far, the variable `daylabel` worked as an index and therefore would only control by trend in a potential regression. This could explain why this variable is ranked as the second most important one using a GBM model, but according to the correlation matrix it is barely correlated with `log_count`. Additionally, we observed in a previous section the necessity of controlling by weekly seasonality, since there is a change in pattern between the different days of the week. That is why we dropped `daylable` and created a new variable called `weekday` to identify the day of the week, going from 1 through 7 starting

on Sunday. Additionall, frmo the matrix we saw that it was important to create dummy variables for month, season and weather for a better understanding of the interaction between their categories.

Once we changed the variables and recoded the values we proceeded to construct our model using all available variables in our training data. We split it into a `training` and a `validating` set, using 25% of our observations for the validation set. We used both a Boosting model and a Random Forest to see which performed better. After tuning and running both models several times, we decided to use the Boosting model which gave use a lower root MSE, and refit the model to predict the bike count for our `test` data set using the following parameters: shrinkage = 0.01, tree depth = 10, minimum number of observations in each terminal node = 5, percent of training data to sample for each tree = 0.77, and optimal number of trees = 909. We analyzed almost 297 different specifications of Boosting model as well as 40 specifications of RF model in this last document. These numbers, however, don't reflect the total number of trials we've made considering different tweaks in the code.

Setup

```
library(data.table)
library(ggplot2)
library(lubridate)
library(ggpubr)
library(viridis)
library(randomForest)
library(pdp)
library(iml)
library(gbm)
library(rpart)
library(xgboost)
library(zoo)
library(corrplot)
library(fastDummies)
library(ranger)
set.seed(1)
```

1 Loading Data

```
setwd("C:/Users/vfuentesc/OneDrive - The University of Chicago/Winter 2021/Machine Learning/Week 3/HW3-ML")
train = data.table(read.csv("Bike_train.csv"))
test = data.table(read.csv("Bike_test.csv"))
```

```
train[, log_count := log(count + 1)]
train[, count := NULL]

season_labels = c("Winter", "Spring", "Summer", "Fall")
train[, season := factor(season, levels = 1:4, labels = season_labels)]
```

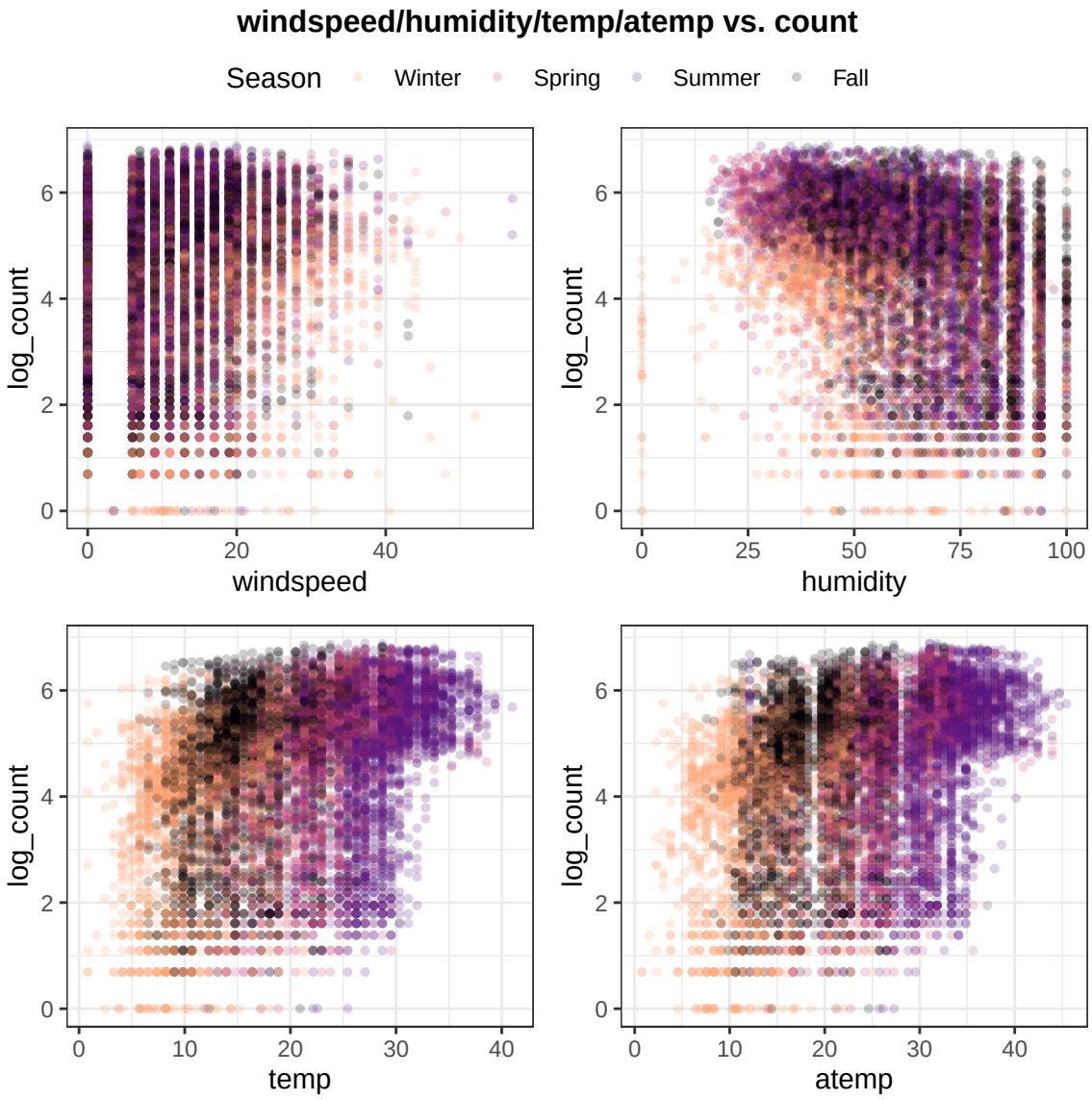
2 Questions

2.1 Data Exploration

a. Visualize the relationship between count and each one of the following variables on a separate scatter plot: `windspeed`, `humidity`, `temp`, and `atemp`.

```
scatter_plot = function(x_var){
return(ggplot(train, aes(x = get(x_var), y = log_count,color = season, fill = season)) +
         geom_point(shape = 21, size = 1.5, stroke = 0.1, alpha = 0.2) +
         scale_fill_viridis_d(option = "A", end = 0.8, direction = -1) + scale_color_viridis_d(option
         labs(x = x_var, fill = "Season", color = "Season") + theme_bw())}
```

```
annotate_figure(ggarrange(scatter_plot("windspeed"), scatter_plot("humidity"),
                          scatter_plot("temp"), scatter_plot("atemp"),
                          common.legend = TRUE),
                top = text_grob("windspeed/humidity/temp/atemp vs. count", face = "bold"))
```

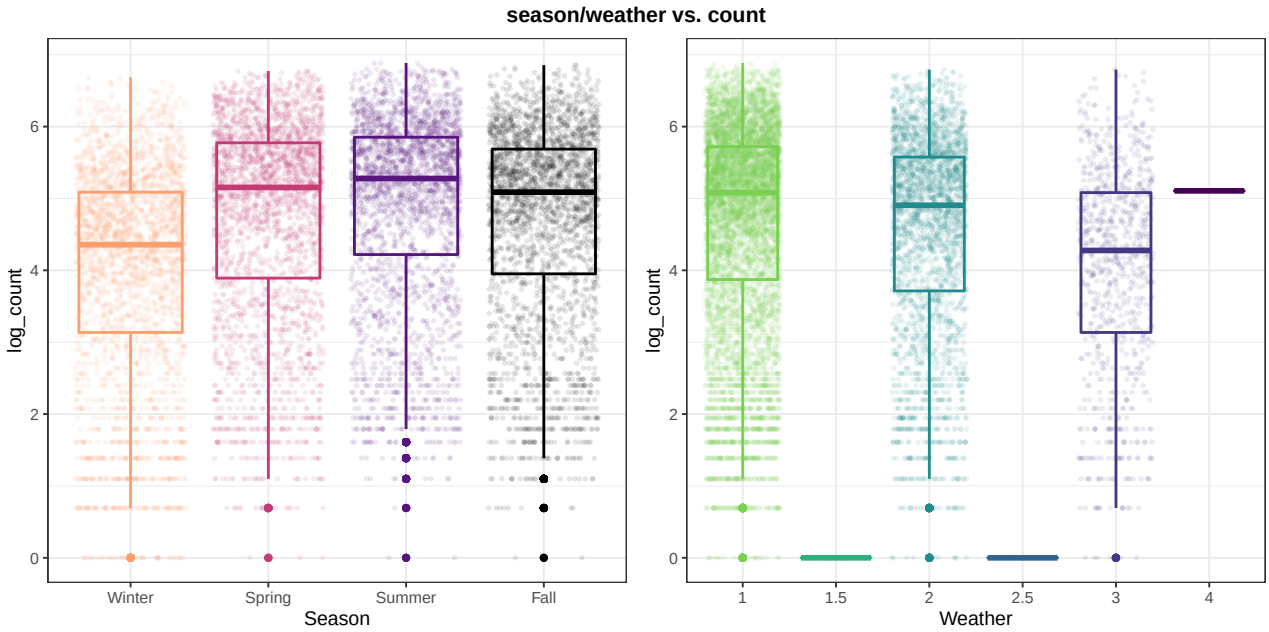


Outliers in ***windspeed*** *(atypical values in Summer),* ***humidity*** *(atypical zero values in Winter),* ***atemp*** *(very low values in Summer).*

b. How does count depend on the season? Consider visualizing this relationship with a boxplot.

```
season_boxplot =
  ggplot(train, aes(x = season, y = log_count)) +
    geom_boxplot(aes(color = season), size = 0.75) +
    geom_jitter(aes(color = season), size = 0.75, alpha = 0.1) +
    scale_color_viridis_d(option = "A", end = 0.8, direction = -1) +
    labs(x = "Season") + theme_bw() + theme(legend.position = "none")

weather_boxplot = ggplot(train, aes(x = factor(weather), y = log_count)) +
    geom_boxplot(aes(color = factor(weather)), size = 0.75) +
    geom_jitter(aes(color = factor(weather)), size = 0.75, alpha = 0.1) +
    scale_color_viridis_d(option = "D", end = 0.8, direction = -1) +
    labs(x = "Weather") + theme_bw() + theme(legend.position = "none")
```

```
annotate_figure(ggarrange(season_boxplot, weather_boxplot),
                top = text_grob("season/weather vs. count", face = "bold"))
```



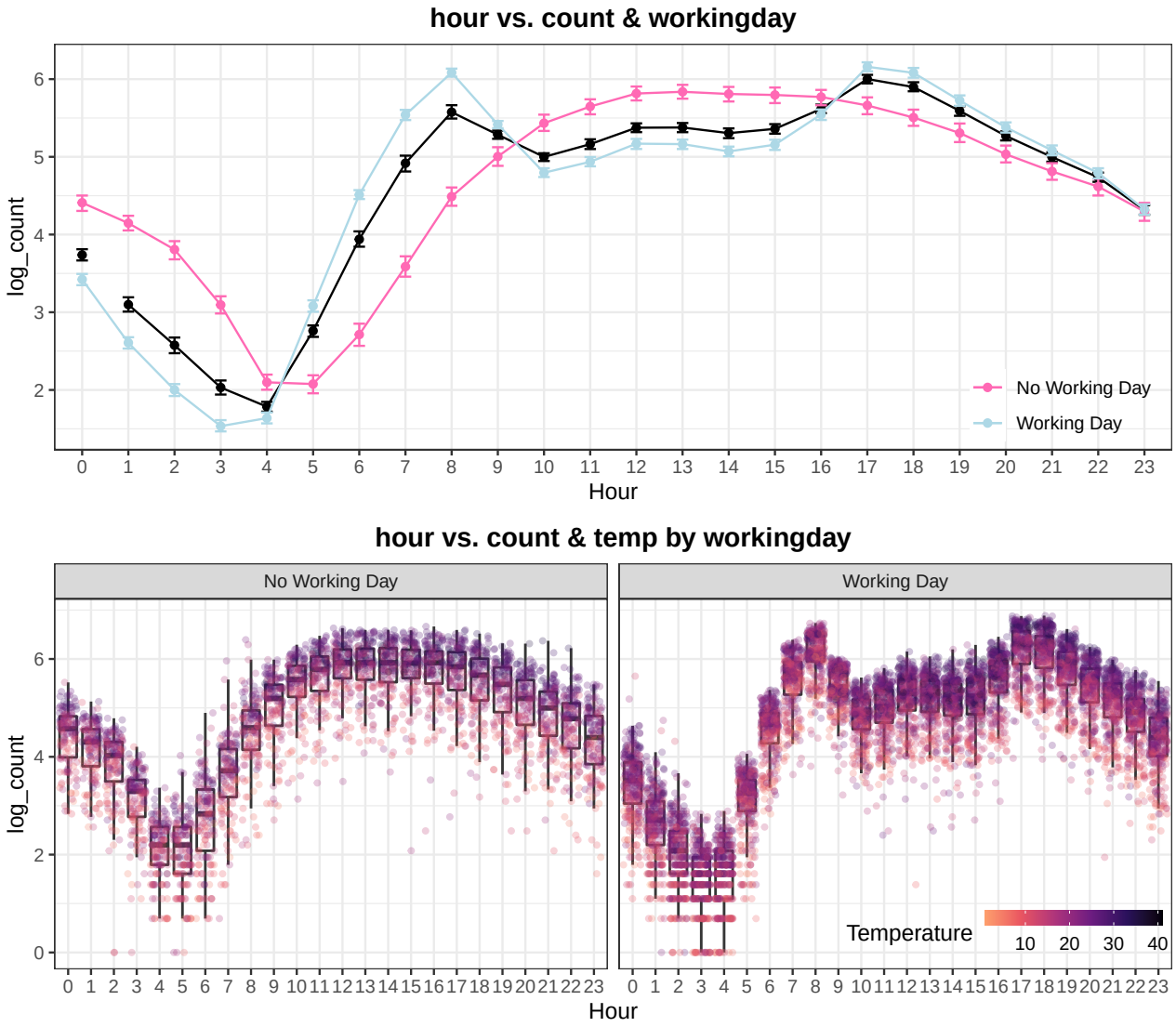
*There is less rentals (**count**) during Winter **season**. On the other hand, **weather** shows irregular values with decimals points.*

c. How does `count` depend on the time of the day (`hour`)? Does this relationship change depending on whether it is a `workingday` or not? You might consider coloring the observations by the temperature (`temp` or `atemp`).

```
hour_wkday_lines =
  ggplot(train, aes(x = factor(hour), y = log_count, group = factor(workingday), color = factor(workin
    stat_summary(fun.y = mean, geom = "line", na.rm = TRUE, group = NA, color = "black") +
    stat_summary(fun.y = mean, geom = "point", na.rm = TRUE, group = NA, color = "black") +
    stat_summary(fun.data = mean_cl_boot, geom = "errorbar", width = 0.25, na.rm = TRUE, group = NA, c
    stat_summary(fun.y = mean, geom = "line") +
    stat_summary(fun.y = mean, geom = "point") +
    stat_summary(fun.data = mean_cl_boot, geom = "errorbar", width = 0.25) +
    scale_color_manual(values = c("hotpink", "lightblue"), labels = c("No Working Day", "Working Day")
    labs(x = "Hour", color = "", title = "hour vs. count & workingday") + theme_bw() +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"), legend.position = c(0.9, 0.15), legen
```

```
hour_wkday_temp_boxplot =
  ggplot(train, aes(x = factor(hour), y = log_count)) +
    geom_boxplot(size = 0.6, color = "gray21", outlier.color = NA) +
    geom_jitter(aes(color = temp), size = 1, alpha = 0.25) +
    scale_color_viridis_c(option = "A", end = 0.8, direction = -1) +
    labs(x = "Hour", color = "Temperature", title = "hour vs. count & temp by workingday") + theme_bw(
    facet_grid(~workingday, labeller = labeller(workingday = c("0" = "No Working Day", "1" = "Working
    theme(plot.title = element_text(hjust = 0.5, face = "bold"), legend.position = c(0.85, 0.10), lege
```

```
ggarrange(hour_wkday_lines, hour_wkday_temp_boxplot, nrow = 2)
```

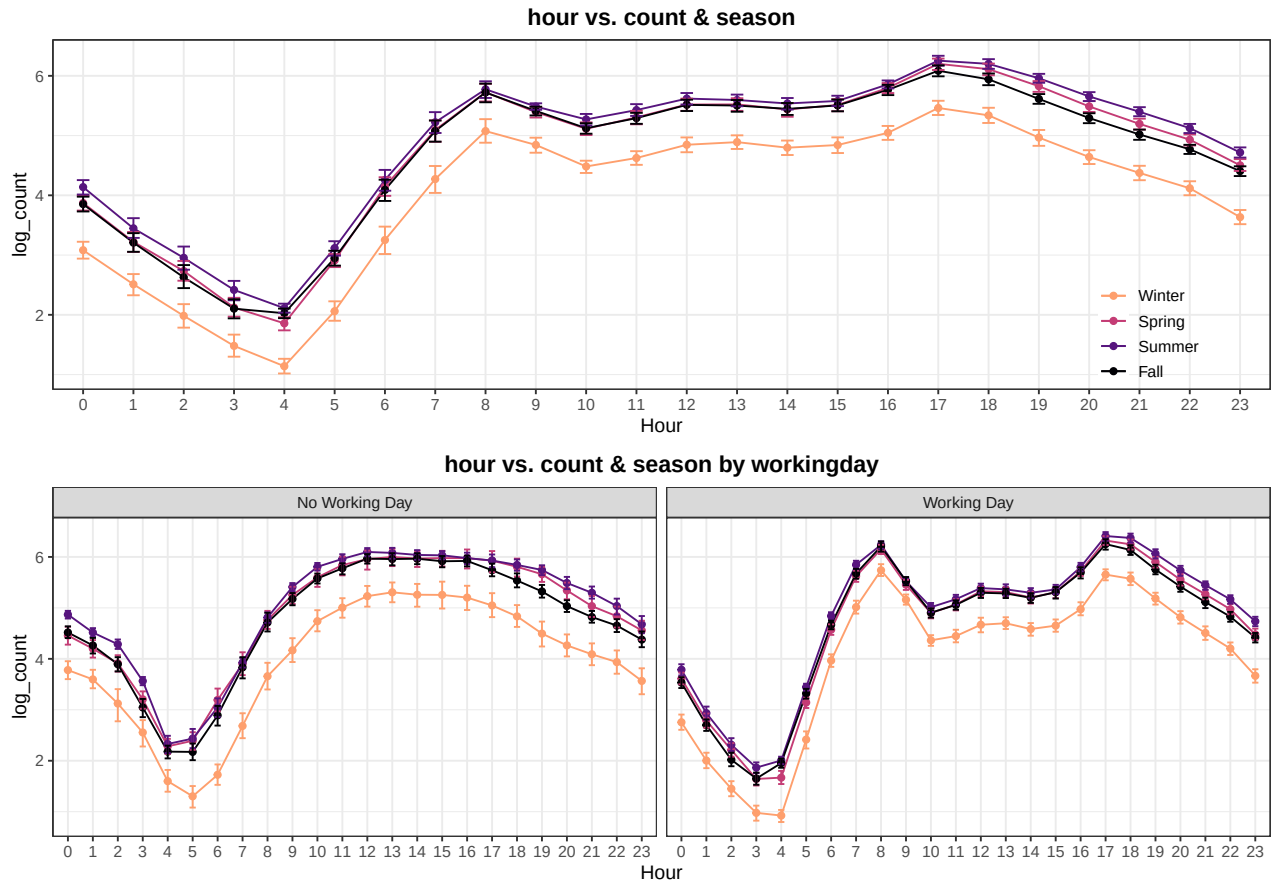


Bike demand higher during peak hours (7-9am and 4-7pm). However, this only remains for working days. During no working days, bike demand is higher from 10am to 6pm. Additionally, irrespective of whether is working day or not, the higher the temperature, the higher the demand for bikes.

d. Does the relationship between `count` and `hour` change by season?

```
hour_season_lines =
  ggplot(train, aes(x = factor(hour), y = log_count, group = season, color = season)) +
    stat_summary(fun.y = mean, geom = "line") +
    stat_summary(fun.y = mean, geom = "point") +
    stat_summary(fun.data = mean_cl_boot, geom = "errorbar", width = 0.25) +
    scale_color_viridis_d(option = "A", end = 0.8, direction = -1) +
    labs(x = "Hour", color = "", title = "hour vs. count & season") + theme_bw() +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"), legend.position = c(0.9, 0.2), legend

hour_season_wkdy_lines =
  ggplot(train, aes(x = factor(hour), y = log_count, group = season, color = season)) +
    stat_summary(fun.y = mean, geom = "line") +
    stat_summary(fun.y = mean, geom = "point") +
    stat_summary(fun.data = mean_cl_boot, geom = "errorbar", width = 0.25) +
    scale_color_viridis_d(option = "A", end = 0.8, direction = -1) +
    labs(x = "Hour", color = "", title = "hour vs. count & season by workingday") + theme_bw() +
    facet_grid(~workingday, labeller = labeller(workingday = c("0" = "No Working Day", "1" = "Working
    theme(plot.title = element_text(hjust = 0.5, face = "bold"), legend.position = "none", legend.back
```

```
ggarrange(hour_season_lines, hour_season_wkdy_lines, nrow = 2)
```

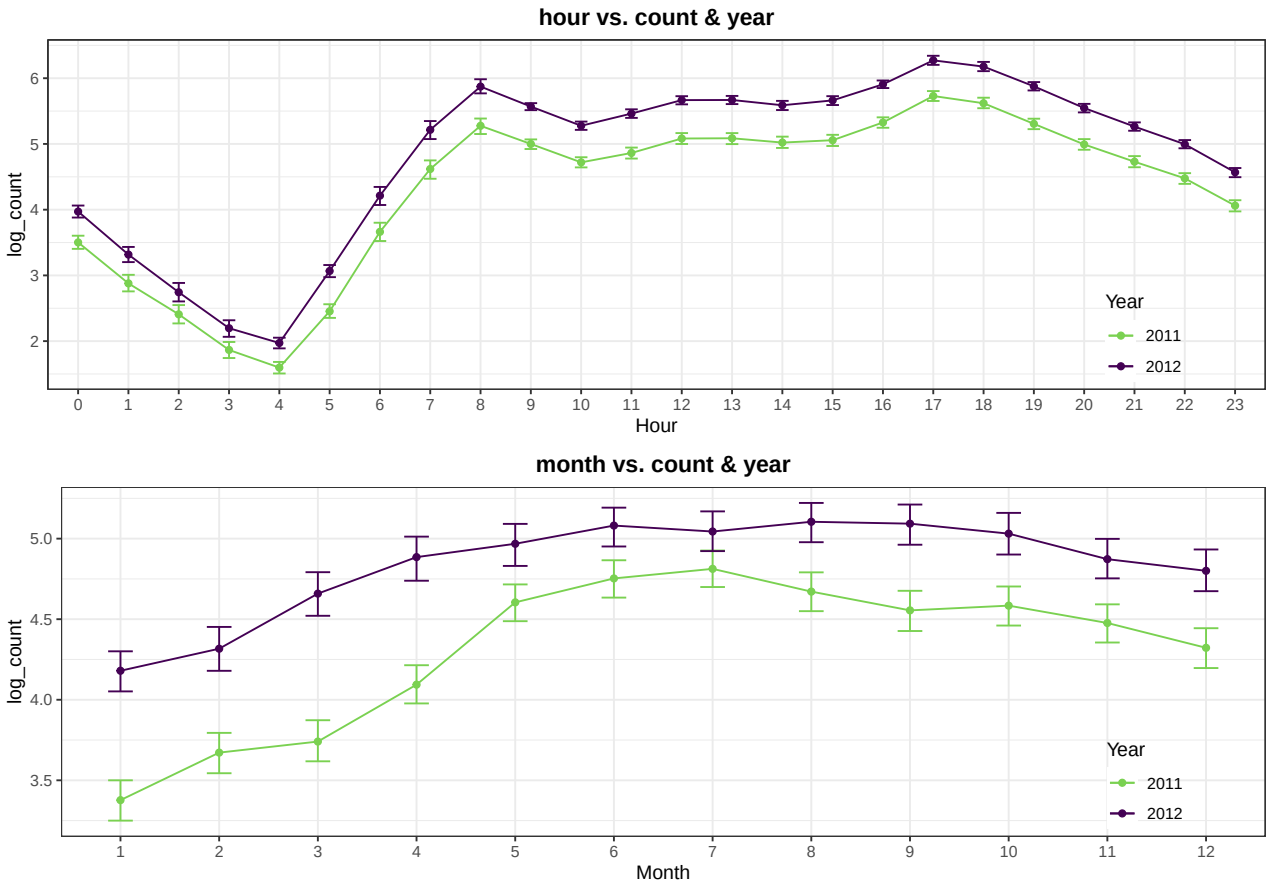


Consistently, there is less demand during Winter. Moreover, demand in Spring/Summer/Fall is almost statistically the same along the day irrespective of whether is a working day or not

e. Does the distribution of hourly number of rentals change between 2011 and 2012? What does this tell you about the rental business?

```
hour_year_lines =
  ggplot(train, aes(x = factor(hour), y = log_count, group = factor(year), color = factor(year))) +
    stat_summary(fun.y = mean, geom = "line") +
    stat_summary(fun.y = mean, geom = "point") +
    stat_summary(fun.data = mean_cl_boot, geom = "errorbar", width = 0.25) +
    scale_color_viridis_d(option = "D", end = 0.8, direction = -1) +
    labs(x = "Hour", color = "Year", title = "hour vs. count & year") + theme_bw() +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"), legend.position = c(0.9, 0.15), legen

hour_year_month_lines =
  ggplot(train, aes(x = factor(month), y = log_count, group = factor(year), color = factor(year))) +
    stat_summary(fun.y = mean, geom = "line") +
    stat_summary(fun.y = mean, geom = "point") +
    stat_summary(fun.data = mean_cl_boot, geom = "errorbar", width = 0.25) +
    scale_color_viridis_d(option = "D", end = 0.8, direction = -1) +
    labs(x = "Month", color = "Year", title = "month vs. count & year") + theme_bw() +
    theme(plot.title = element_text(hjust = 0.5, face = "bold"), legend.position = c(0.9, 0.15), legen
```

```
ggarrange(hour_year_lines, hour_year_month_lines, nrow = 2)
```



Yes, it changes. Rental business has expanded between 2011 and 2012.

2.2 Fitting a Random Forest model and a Boosting model

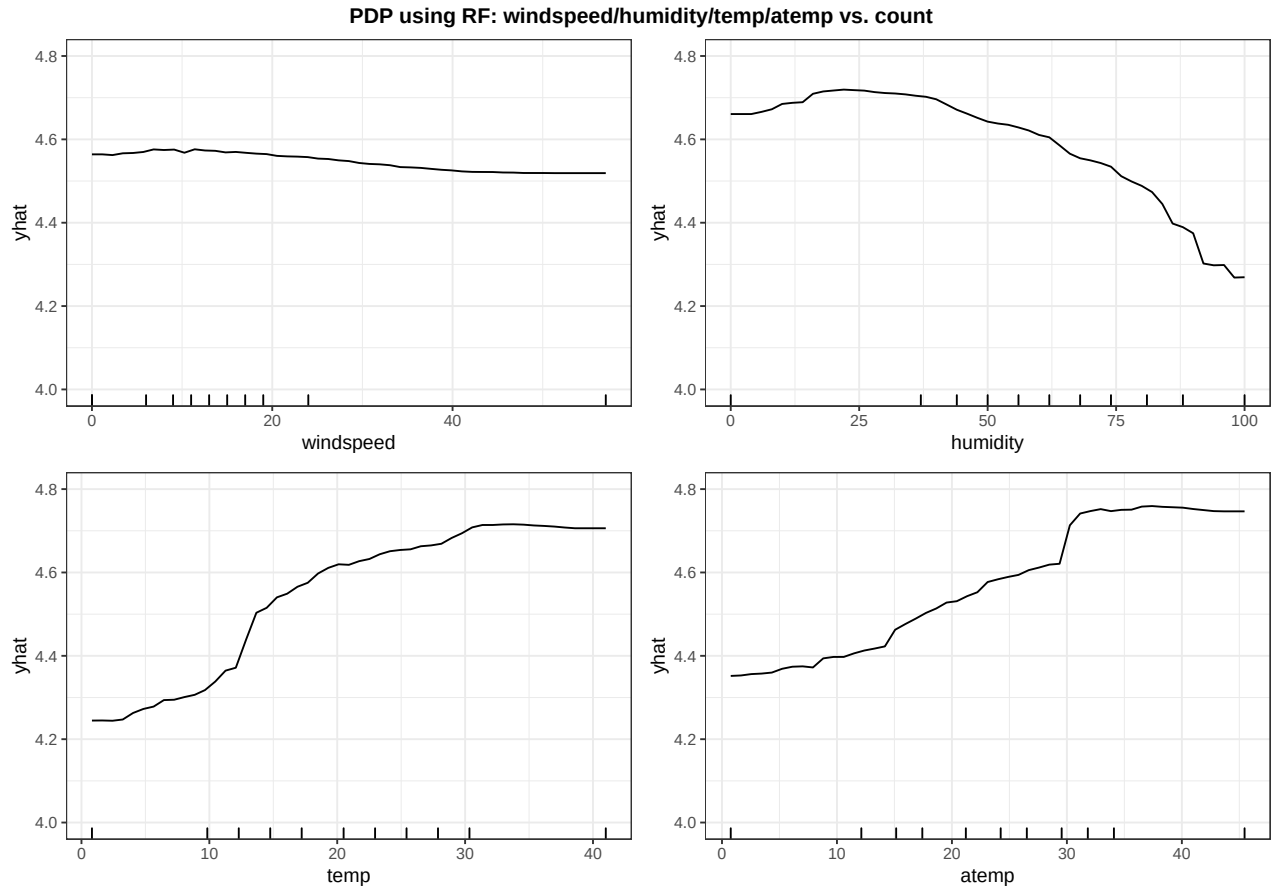
a. Create a partial dependence plot for predicted `count` vs each one of the following variables: `windspeed`, `humidity`, `temp`, and `atemp`. Do this for both the *Random Forest* and the *Boosting* model.

```
pdp_plot = function(model, pred_var, ylim, n.trees = NULL){
  if (as.character(substitute(model)) == "fit_rf") {
  temp = autoplot(partial(model, pred_var), rug = T, train = train)
  } else if (as.character(substitute(model)) == "fit_gbm") {
  temp = autoplot(partial(model, pred_var, n.trees = n.trees), rug = T, train = train)
  } else{
  return(print("No recognized model"))
  }
  return(temp + scale_y_continuous(limits = ylim) + theme_bw())
}
```

Random Forest model

```
fit_rf = randomForest(log_count ~ ., data = train, mtry = 4,
                      ntree = 1000, nodesize = 25, keep.inbag = TRUE, importance = TRUE)
```

```
ylim = c(4.0, 4.8)
annotate_figure(ggarrange(pdp_plot(fit_rf, "windspeed", ylim), pdp_plot(fit_rf, "humidity", ylim),
                          pdp_plot(fit_rf, "temp", ylim), pdp_plot(fit_rf, "atemp", ylim)),
                top = text_grob("PDP using RF: windspeed/humidity/temp/atemp vs. count", face = "bold"
```



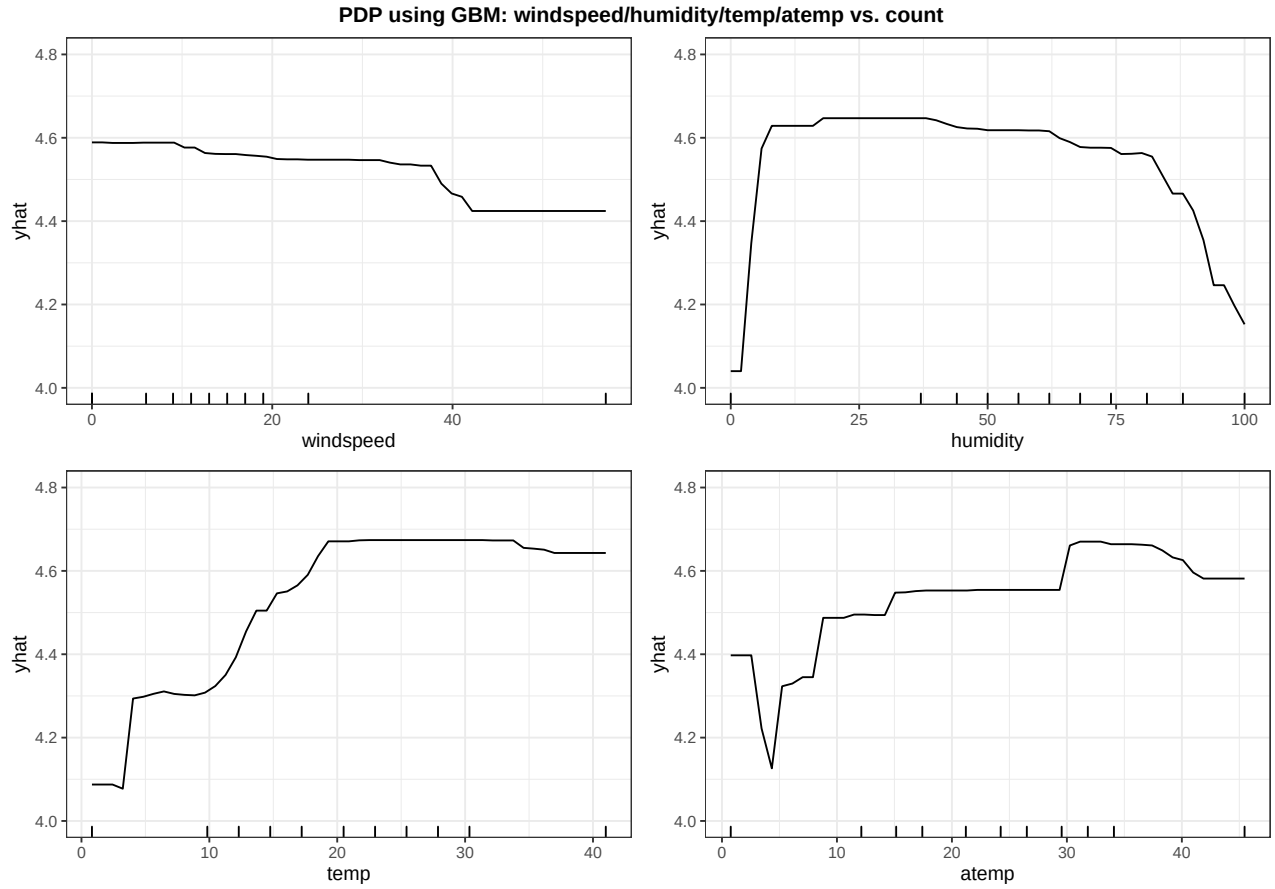
Using a simple Random Forest model with ***ntree*** *of 1,000 and* ***mtry*** *of 4, we find that* ***windspeed*** *almost doesn't predict* ***yhat****, while* ***humidity*** *shows a negative relationship. Moreover, both* ***temp*** *and* ***atemp*** *present a positive relationship with* ***yhat*** *below ~30 Celsius degress.*

Boosting model

```
fit_gbm = gbm(log_count ~ ., data = train, distribution = "gaussian",
              interaction.depth = 1, n.trees = 5000, shrinkage = 0.01,
              cv.folds = 5, n.cores = 1, verbose = FALSE)
```

```
## CV: 1
## CV: 2
## CV: 3
## CV: 4
## CV: 5
```

```
n_trees = 5000
annotate_figure(ggarrange(pdp_plot(fit_gbm, "windspeed", ylim, n_trees), pdp_plot(fit_gbm, "humidity",
                          pdp_plot(fit_gbm, "temp", ylim, n_trees), pdp_plot(fit_gbm, "atemp", ylim, n
                top = text_grob("PDP using GBM: windspeed/humidity/temp/atemp vs. count", face = "bold
```

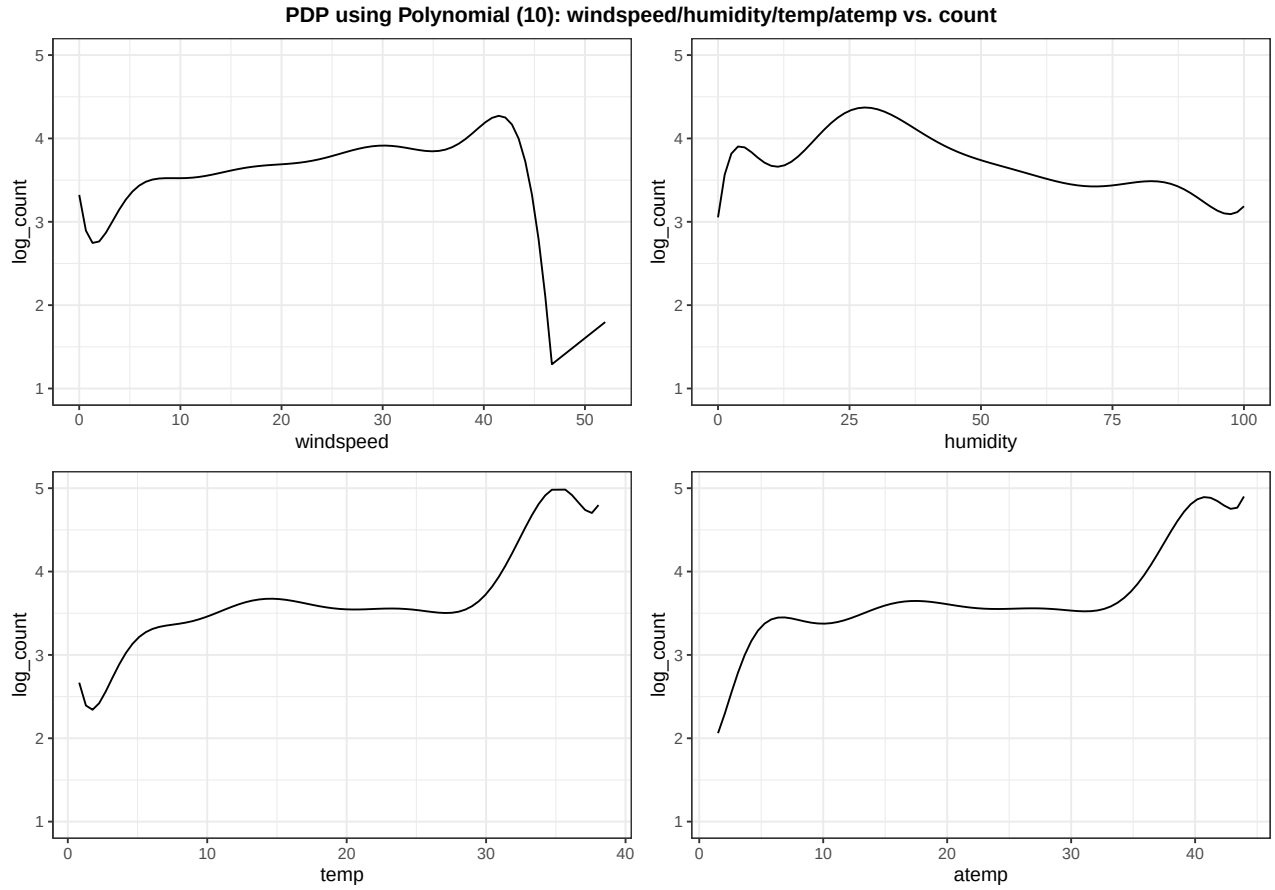


GBM: Explanation

b. Build a marginal model that regresses count on each one of the following variables: `windspeed`, `humidity`, `temp`, and `atemp`. You can use whichever nonlinear model you want for these regression tasks. Plot the marginal fits and compare them with partial dependence plots above. How are the plots different and why?

```
poly_plot = function(x_var){
return(ggplot(train, aes(x = get(x_var), y = log_count)) +
         geom_smooth(formula = y ~ poly(x, 10), se = FALSE, color = "black", size = 0.5) +
         scale_y_continuous(limits = c(1, 5)) +
         labs(x = x_var) + theme_bw())
}
```

```
annotate_figure(ggarrange(poly_plot("windspeed"), poly_plot("humidity"),
                          poly_plot("temp"), poly_plot("atemp")),
                top = text_grob("PDP using Polynomial (10): windspeed/humidity/temp/atemp vs. count",
```

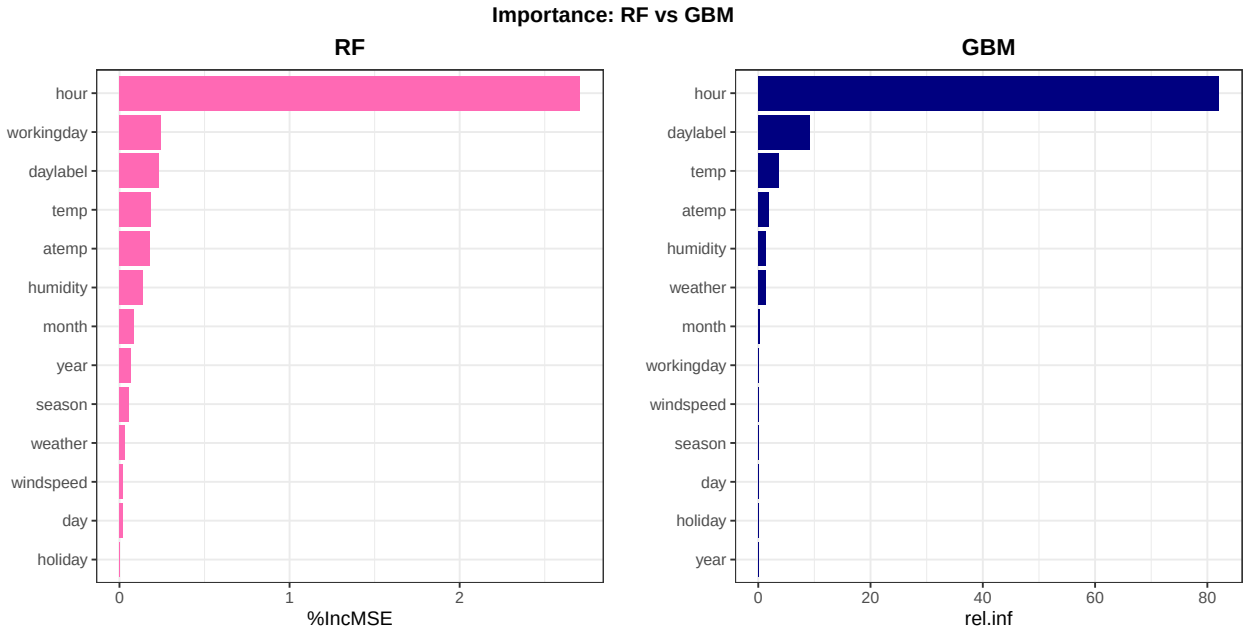


Analysis. A model with a single regressor is so poor

c. Create variable importance plots for the two models in part 2.a. Do the two models rank the variables in the same way?

```
importance_rf = as.data.table(fit_rf$importance, keep.rownames = T)
importance_gbm = as.data.table(summary(fit_gbm))
```

```
importance_features = list(coord_flip(), theme_bw(),
                           theme(plot.title = element_text(hjust = 0.5, face = "bold")))

rf_importance_plot = ggplot(importance_rf, aes(x = reorder(rn, `%IncMSE`), y = `%IncMSE`)) +
                            geom_col(fill = "hotpink") + labs(x = "", title = "RF") +
                            importance_features

gbm_importance_plot = ggplot(importance_gbm, aes(x = reorder(var, rel.inf), y = rel.inf)) +
                            geom_col(fill = "navyblue") + labs(x = "", title = "GBM") +
                            importance_features

annotate_figure(ggarrange(rf_importance_plot, gbm_importance_plot),
                top = text_grob("Importance: RF vs GBM", face = "bold"))
```



RF and GBM rank the same the first variable: ***hour****. While RF includes* ***workingday*** *as the second best predictor, GBM ranks it as the seventh. Later, both models ranks* ***daylabel****,* ***atemp****, and* ***temp*** *in the same order.*

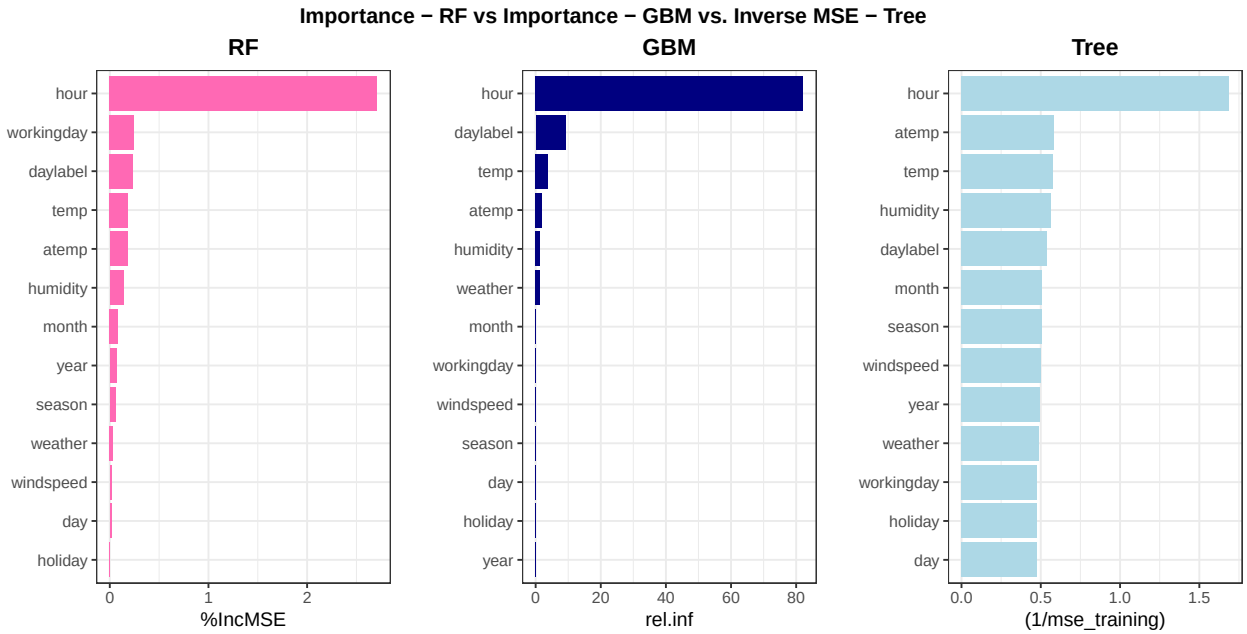
2.3 Investigate how predictive each variable is on its own.

There are a number of ways to do this. The simplest way would be to regress `count` on each one of the variables separately and evaluate the out-of-sample MSE for each one of the models. For instance, you could fit a regression tree model.

a. What can you say by comparing how predictive each variable is on its own vs the variable importance ranking obtained in the previous question?

```
MSE_tree_models = data.table(n = 1:length(importance_gbm$var))

for (i in 1:length(importance_gbm$var)) {
  model_of_use = paste("log_count", importance_gbm$var[i], sep = " ~ ")


  tree = rpart(as.formula(model_of_use),
               data = train,
               method = "anova",
               control = rpart.control(minsplit = 5, cp = 0.0001, xval = 10))

  N_terminal_nodes = length(unique(tree$where))
  index_optimal = which.min(tree$cptable[, "xerror"])
  cp_optimal = tree$cptable[index_optimal, "CP"]
  pruned_tree = prune(tree, cp = cp_optimal)
  N_optimal_nodes = length(unique(pruned_tree$where))

  MSE_tree_models[n == i, model := importance_gbm$var[i]]
  MSE_tree_models[n == i, nodes_original := N_terminal_nodes]
  MSE_tree_models[n == i, nodes_optimal := N_optimal_nodes]
  MSE_tree_models[n == i, mse_training := mean((train$log_count - predict(pruned_tree, train))^2)]
}
```

```
cv_mse_plot = ggplot(MSE_tree_models, aes(x = reorder(model, (1/mse_training)), y = (1/mse_training)))
                geom_col(fill = "lightblue") + labs(x = "", title = "Tree") +
                importance_features

annotate_figure(ggarrange(rf_importance_plot, gbm_importance_plot, cv_mse_plot, nrow = 1),
                top = text_grob("Importance - RF vs Importance - GBM vs. Inverse MSE - Tree", face = "
```



- *Similarities:* `hour`, `temp` *and* `atemp` *are consistently well ranked across all three instruments. In the opposite,* `holiday` *and* `day` *seem bad candidates to predictor.*
- *Differences:* `workingday` *is only well ranked under RF model.*

b. Why do you think is the reason for the difference?

- *There are interactions between explanatory variables:* `workingday` *and* `hour`*.*
- *There are high correlations between explanatory variables:* `temp` *and* `atemp`*.*
- *There are seasonality in data: weekly pattern not controlled yet.*

c. How could you use the variable importance ranking to select variables? We will talk about this in detail in Week 5. Here I want you to think a bit about the variable selection problem and how would you use the tools that you have learnt so far to identify a good set of variables.

We could start with the simplest model including only the most important variable and then recursively include other variables. We should pick up a set of variables that are correlated with `log_count` *as well as it doesn't repeat information could harm the model.*

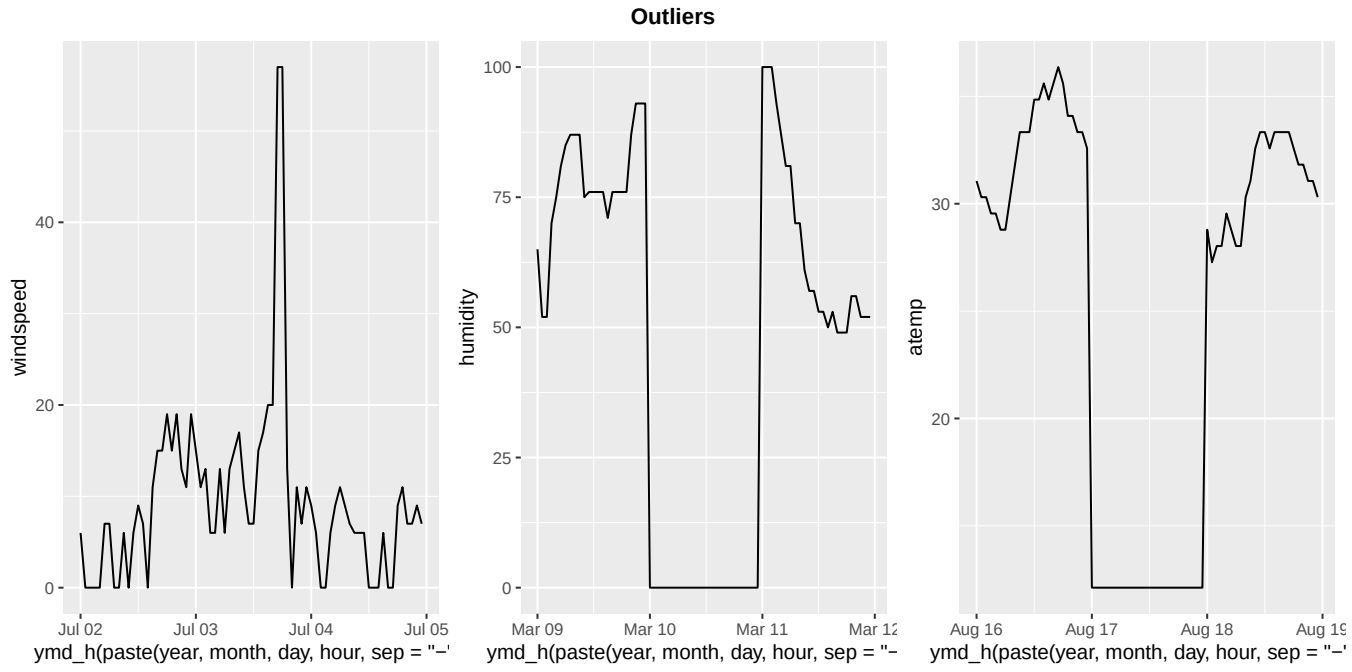
2.4 Build a model to predict the bikeshare counts

2.4.1 Outliers and Weird Numbers

We started by examining the data graphically in section 2. We added a color aesthetic by season to the scatter plots between `count` and `windspeed`, `humidity`, `atemp`, and `temp`. Then, we noted outliers in the first three variables (Section 2.1.a). In particular, we observed twice `windspeed` repeated with the value of 56.9969. Moreover, we also realized that during Winter there was one day, March 10, with humidity equal to zero, which is not possible. Additionally, we found an outlier in `atemp`, a very low temperature in the Summer. We further explored these outliers below to identify the exact date and time and replaced the outliers. For `windspeed`, we interpolated with the closest numbers, while for for `atemp` and `humidity`, we used an average between values by hours from the day before and the day after.

We continued exploring dataset by creating boxplots for both `season` and `weather` (Section 2.1.b). There, we found that in `weather` there were some half values, 1.5 and 2.5, which did not fall into any specific category. Using a conservative approach, we changed these values by rounding up to the nearest number with decimals so they can be representative of one of the four weather categories

```
outlier_windspeed = train[daylabel %in% c(183:185), c("daylabel", "year", "month", "day", "hour", "windspee
outlier_humidity = train[daylabel %in% c(68:70), c("daylabel", "year", "month", "day", "hour", "humidity")]
outlier_atemp = train[daylabel %in% c(594:596), c("daylabel", "year", "month", "day", "hour", "atemp")]

annotate_figure(ggarrange(ggplot(outlier_windspeed, aes(x = ymd_h(paste(year, month, day, hour, sep = "-"))
                          ggplot(outlier_humidity, aes(x = ymd_h(paste(year, month, day, hour, sep = "-")),
                          ggplot(outlier_atemp, aes(x = ymd_h(paste(year, month, day, hour, sep = "-")), y
                top = text_grob("Outliers", face = "bold"))
```



- For `windspeed`: Interpolating with the closest numbers

```
train[daylabel == 184 & hour %in% c(17, 18), c("windspeed")] = NA
train[, windspeed := na.approx(train$windspeed)]
```

- For `humidity` and `atemp`: Making average between values by hours from the day before and the day after

```
for (i in 0:23) {
  train[daylabel == 69 & hour == i, c("humidity")] =
                  (train[daylabel == 68 & hour == i, c("humidity")] +
                   train[daylabel == 70 & hour == i, c("humidity")]) / 2

  train[daylabel == 595 & hour == i, c("atemp")] =
                   (train[daylabel == 596 & hour == i, c("atemp")] +
                    train[daylabel == 597 & hour == i, c("atemp")]) / 2
}
```

- For `weather`: Rounding up thouse numbers with decimals since they are trying to depict a change in weather sensation.

```
train[, weather := round(weather, 0)]
```

2.4.2 Correcting `holiday/workingday`

After several trial and error attempts, we also found that some days could be better identify as what people perceive in reality over what federal calendar indicates. In particular, we noticed an interesting pattern about Tax Day. When it's originally assigned on Fridays, it's moved to the next Monday. In this case, we appreciated that April 15th 2011 (Friday) and April 16th in 2012 (Friday) show a pattern like a regular `workingday`, while Monday 18th and Monday 19th show a pattern similar to a `holiday`. Hence, we recoded those days as `workingday = 1` and `holiday = 0`.

Under this approach, we also recoded Friday after Thanksgiving, the Christmas Eve and the New Year Eve. We changed `workingday = 0` and `holiday = 1` in the `test` set.

- For Tax Day

```
train[year == 2011 & month == 4 & day == 15, "workingday"] = 1   #2011
train[year == 2011 & month == 4 & day == 15, "holiday"] = 0
train[year == 2012 & month == 4 & day == 16, "workingday"] = 1   #2012
train[year == 2012 & month == 4 & day == 16, "holiday"] = 0
```

- For Thanksgiving Friday

```
test[year == 2011 & month == 11 & day == 25, "workingday"] = 0   #2011
test[year == 2011 & month == 11 & day == 25, "holiday"] = 1
test[year == 2012 & month == 11 & day == 23, "workingday"] = 0   #2012
test[year == 2012 & month == 11 & day == 23, "holiday"] = 1
```

- For Christmas Eve and New Year Eve

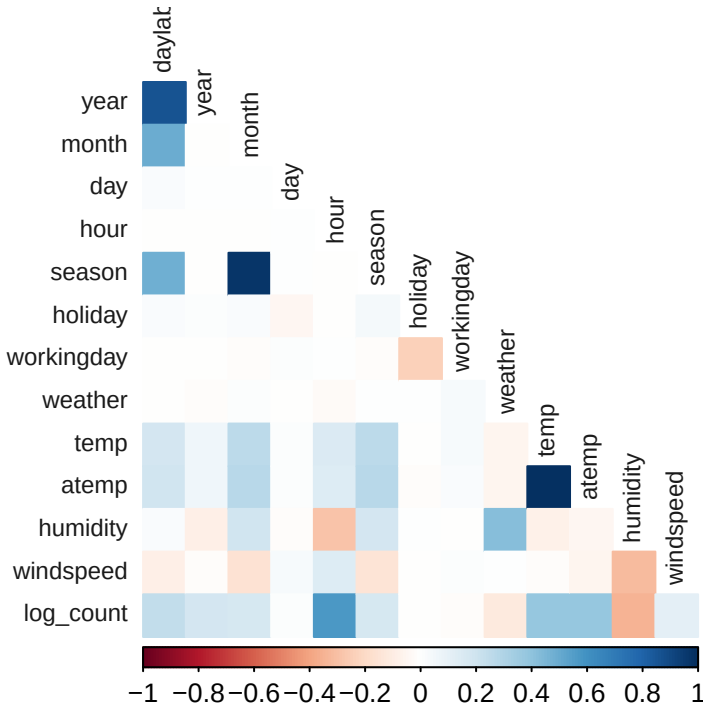
```
test[year == 2012 & month == 12 & day == 24, "workingday"] = 0   #2012
test[year == 2012 & month == 12 & day == 24, "holiday"] = 1
test[year == 2012 & month == 12 & day == 31, "workingday"] = 0   #2012
test[year == 2012 & month == 12 & day == 31, "holiday"] = 1
```

2.4.3 Correlation Matrix

After recoding the variables in our `train` and `test` data set we further explored the interaction and importance of our variables. We did this by constructing Partial Dependence Plots using a Random Forest model, a Boosting mode, individual Tree Regressions (`log_count ~ $var_i$`), and using a correlation matrix. So far, the variable `daylabel` worked as an index and therefore would only control by trend in a potential regression. This could explain why this variable is ranked as the second most important one using a GBM model, but according to the correlation matrix it

is barely correlated with `log_count`. Additionally, we observed in a previous section the necessity of controlling by weekly seasonality, since there is a change in pattern between the different days of the week. That is why we dropped `daylable` and created a new variable called `weekday` to identify the day of the week, going from 1 through 7 starting on Sunday. Additionall, frmo the matrix we saw that it was important to create dummy variables for month, season and weather for a better understanding of the interaction between their categories.

```
train_cor = copy(train)
train_cor[, season := as.numeric(season)]
corrplot(cor(train_cor), method = "color", type = "lower", diag = FALSE, tl.cex = 0.75, tl.col = "gray10")
```



2.4.4 The necessity of new variables

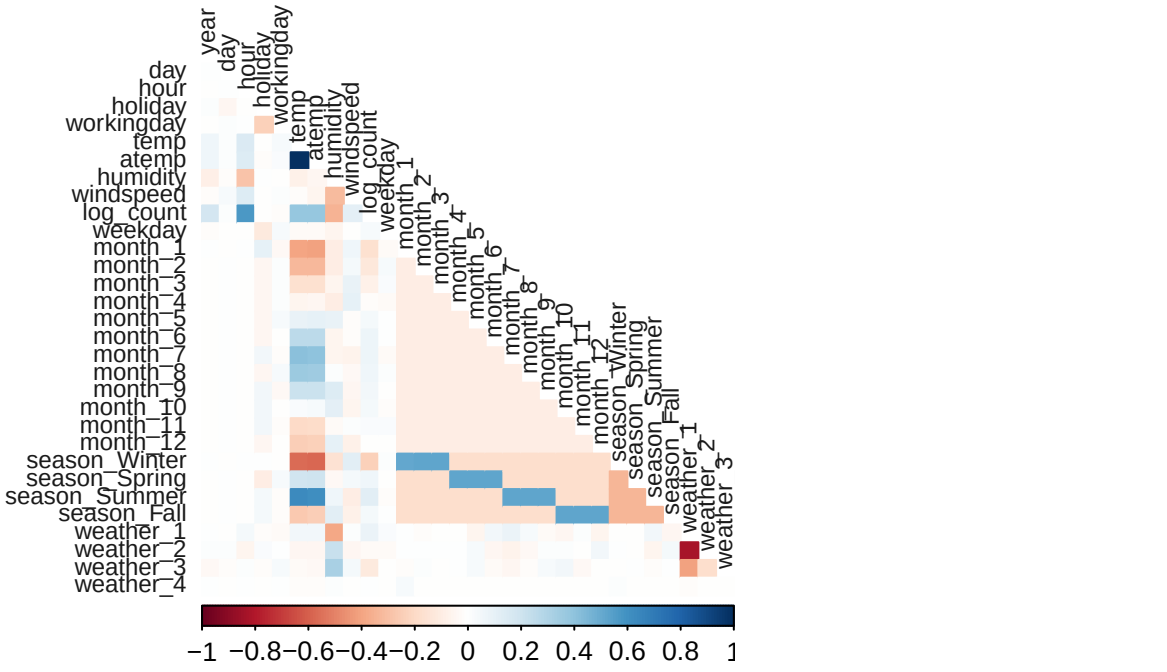
So far, `daylabel` is working as index, thus it would control by trend in a potential regression. This could explain why this variable is ranked as the second most important one using a GBM model, but according to the correlation matrix it is barely correlated with `log_count`. Additionally, we observed in a previous section the necessity of controlling by weekly seasonality. In this regard, we will change the `daylabel` variable for `weekday` variable.

```
train[, weekday := wday(ymd(paste(year, month, day, sep = "-")))]
train[, daylabel := NULL]
```

Additionally, we will transform `month`, `season` and `weather` into dummies variables

```
train = dummy_cols(train, select_columns = "month", remove_selected_columns = T)
train = dummy_cols(train, select_columns = "season", remove_selected_columns = T)
train = dummy_cols(train, select_columns = "weather", remove_selected_columns = T)
```

```
train_cor = copy(train)
corrplot(cor(train_cor), method = "color", type = "lower", diag = FALSE, tl.cex = 0.75, tl.col = "gray10")
```



2.4.5 Engineering models

Once we changed the variables and recoded the values we proceeded to construct our model using all available variables in our training data. We split it into a `training` and a `validating` set, using 25% of our observations for the validation set. We used both a Boosting model and a Random Forest to see which performed better. After tuning and running both models several times, we decided to use the Boosting model which gave use a lower root MSE, and refit the model to predict the bike count for our `test` data set using the following parameters: shrinkage = 0.01, tree depth = 10, minimum number of observations in each terminal node = 5, percent of training data to sample for each tree = 0.77, and optimal number of trees = 909.

- Splitting data: We will divide the `train` dataset into `training` (75%) and `validating` (25%)

```
prop_training = 0.75
training_index = sample(nrow(train), prop_training * nrow(train))

training = train[training_index,]
validating = train[-training_index,]
```

2.4.5.1 Boosting model

- Hyperparapter Grid

```
hyper_grid <- expand.grid(
  shrinkage = c(.01, .1, .3),     ## controls the learning rate
  interaction.depth = c(3, 5, 10), ## tree depth
  n.minobsinnode = c(5, 10, 30), ##  minimum number of observations required in each terminal node
  bag.fraction = seq(.7, .8, .01),  ##  percent of training data to sample for each tree
  optimal_trees = 0,               # a place to dump results
  min_RMSE = 0                     # a place to dump results
)
```

- Search of best model

```
X.train = as.matrix(training[,!c("log_count")], with = FALSE)
Y.train = training$log_count

for(i in 1:nrow(hyper_grid)) {

  # create parameter list
  params <- list(
    eta = hyper_grid$shrinkage[i],
    max_depth = hyper_grid$interaction.depth[i],
    min_child_weight = hyper_grid$n.minobsinnode[i],
    subsample = hyper_grid$bag.fraction[i]
  )

  # reproducibility
  set.seed(1)

  # train model
  xgb.tune <- xgb.cv(
    params = params,
    data = X.train,
    label = Y.train,
    nrounds = 3000,
    nfold = 5,
    objective = "reg:squarederror",     # for regression models
    verbose = 0,                        # silent,
    early_stopping_rounds = 10          # stop if no improvement for 10 consecutive trees
  )

  # add min training error and trees to grid
  hyper_grid$optimal_trees[i] <- which.min(xgb.tune$evaluation_log$test_rmse_mean)
  hyper_grid$min_RMSE[i] <- min(xgb.tune$evaluation_log$test_rmse_mean)
}
```

- Piking the best set of parameter

```
oo = hyper_grid[order(hyper_grid$min_RMSE),]
```

- Developing final model

```
# parameter list
params <- list(
  eta = oo[1,]$shrinkage,
  max_depth = oo[1,]$interaction.depth,
  min_child_weight = oo[1,]$n.minobsinnode,
  subsample = oo[1,]$bag.fraction
)

# train final model
xgb.fit.final <- xgboost(
  params = params,
  data = X.train,
  label = Y.train,
  nrounds = oo[1,]$optimal_trees,
  objective = "reg:squarederror",
  verbose = 0
)
```

- Calculating validating-MSE

```
yhat.xgb <- predict(xgb.fit.final, newdata = as.matrix(validating[,!c("log_count")], with = FALSE))
mean((validating$log_count - yhat.xgb) ^ 2)
```

```
## [1] 0.08196366
```

2.4.5.2 Random Forest model

- Hyperparapter Grid and Searching best model

```
hyper_grid_rf <- expand.grid(
  mtry       = seq(9, 30, by = 3),
  node_size  = c(25, 50, 100, 150, 200),
  OOB_RMSE   = 0
)


for(i in 1:nrow(hyper_grid_rf)) {

  # reproducibility
  set.seed(1)

  # train model
  model <- ranger(
    formula         = log_count ~ .,
    data            = training,
    num.trees       = 500,
    mtry            = hyper_grid_rf$mtry[i],
    min.node.size   = hyper_grid_rf$node_size[i],
    seed            = 345
  )
```

```
  # add OOB error to grid
  hyper_grid_rf$OOB_RMSE[i] <- sqrt(model$prediction.error)
}
```

- Piking the best set of parameter

```
oo_rf = hyper_grid_rf[order(hyper_grid_rf$OOB_RMSE),]
```

- Developing final model

```
rf.fit.final <- ranger(
    formula         = log_count ~ .,
    data            = training,
    num.trees       = 500,
    mtry            = oo_rf[1,]$mtry,
    min.node.size   = oo_rf[1,]$node_size
    )
```

- Calculating validating-MSE

```
yhat.rf = predict(rf.fit.final, data = validating)$predictions
mean((validating$log_count - yhat.rf) ^ 2)
```

```
## [1] 0.1199512
```

2.4.6 Adecuating test dataset

```
test[, weekday := wday(ymd(paste(year, month, day, sep = "-")))]
test[, daylabel := NULL]
test[, season := factor(season, levels = 1:4, labels = season_labels)]
test = dummy_cols(test, select_columns = "month", remove_selected_columns = T)
test = dummy_cols(test, select_columns = "season", remove_selected_columns = T)
test = dummy_cols(test, select_columns = "weather", remove_selected_columns = T)
```

2.4.7 Calculating log_count and recoverting to count

```
yhat.final <- predict(xgb.fit.final, newdata = as.matrix(test))
count_hat <- exp(yhat.final) - 1
```

3 Appendix

```
sampleSubmission = data.table(Id = 1:length(count_hat),
                              count = round(count_hat, 0))

write.csv(sampleSubmission,
          file = "sampleSubmission.csv",
          row.names = FALSE,
          quote = FALSE)
```