# ST 563 Final Project - Bike Sharing Data

Team #1

Peter Lung, David Shaw, Yijia Cai

11/4/2021

## Contents

## Introduction

For our project, we are analyzing the bike sharing dataset from the UCI Machine Learning database. The data is the number of daily bike rentals (registered plus "casual") over a two year period. Predictors for this model include variables pertaining to seasonality and weather.

The following is the description of the data from UCI's ML database:

*Bike sharing systems are new generation of traditional bike rentals where whole process from membership, rental and return back has become automatic. Through these systems, user is able to easily rent a bike from a particular position and return back at another position. Currently, there are about over 500 bike-sharing programs around the world which is composed of over 500 thousands bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues.*

*Apart from interesting real world applications of bike sharing systems, the characteristics of data being generated by these systems make them attractive for the research. Opposed to other transport services such as bus or subway, the duration of travel, departure and arrival position is explicitly recorded in these systems. This feature turns bike sharing system into a virtual sensor network that can be used for sensing mobility in the city. Hence, it is expected that most of important events in the city could be detected via monitoring these data.*

The response variable from this dataset is *cnt* which is the total daily count of bike renters. Two other response variables are present in the dataset: *registered* and *casual*. The response we have chosen is *cnt*, which is the sum of the other two.

Our interest is twofold:

(1) Find the best function of the variable set for predicting the response

(2) Evaluate several candidate modeling types for variable selection and data reduction

The predictor variables all pertain to either weather conditions or seasonal effects, some of which can be highly collinear. That makes this data a great set for testing variable selection methods including forward, backward, best subsets, lasso and ridge. It also makes this data a great candidate for testing data reduction methods such as principal components analysis.

We are also interested in testing which variables have nonlinear effects and modeling them with splines. In all tests, we will evaluate the models with cross validation and test each model's predictive power with a consistent holdout sample.

## Methods

This section will detail the methodologies and evaluations used for each type of model tested.

### Exploratory Data Analysis

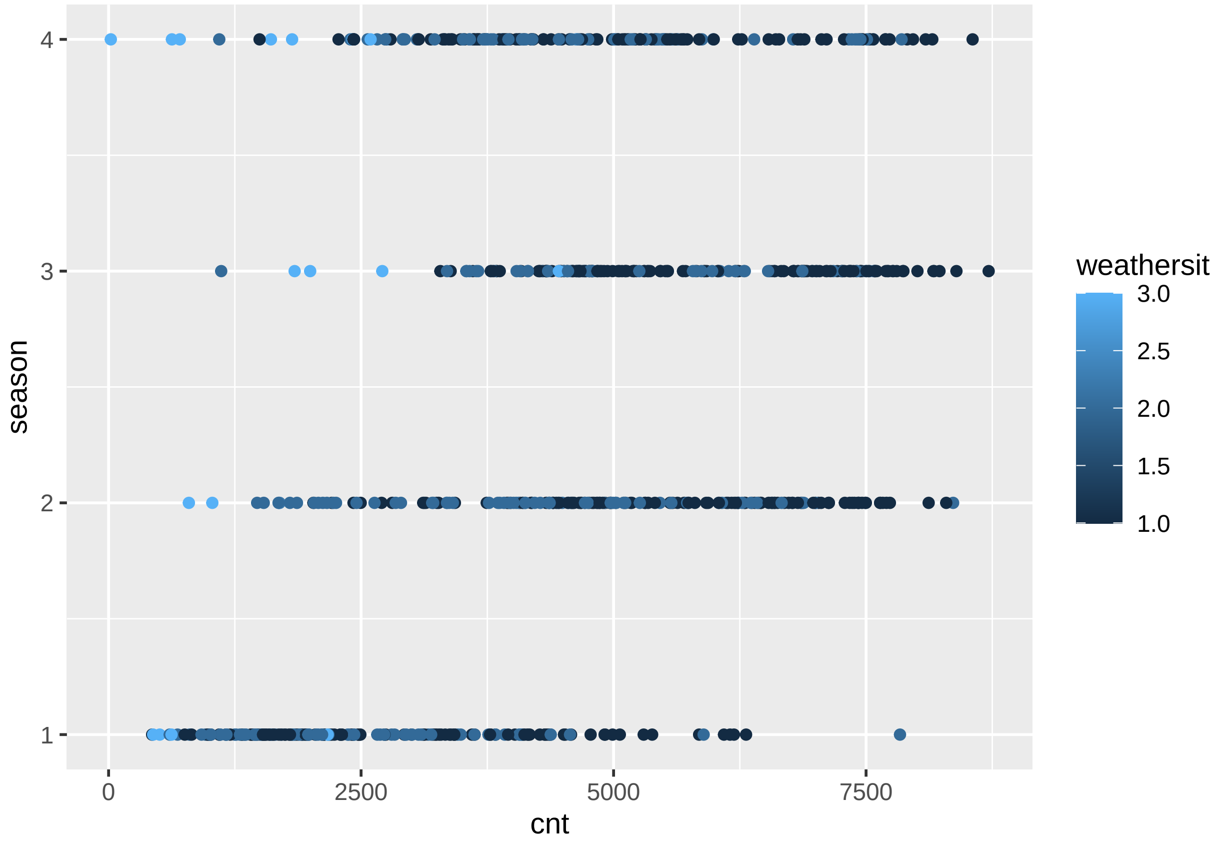
```
#Required: install remotes package
#install.packages("remotes")
# install corrmorant from the github repository
#remotes::install_github("r-link/corrmorant")

#Check out the dataset
head(day)
```

```
## # A tibble: 6 x 14
##   instant dteday     season    yr  mnth holiday weekday workingday weathersit
##     <dbl> <date>      <dbl> <dbl> <dbl>   <dbl>   <dbl>      <dbl>      <dbl>
## 1       1 2011-01-01      1     0     1       0       6          0          2
## 2       2 2011-01-02      1     0     1       0       0          0          2
## 3       3 2011-01-03      1     0     1       0       1          1          1
## 4       4 2011-01-04      1     0     1       0       2          1          1
## 5       5 2011-01-05      1     0     1       0       3          1          1
## 6       6 2011-01-06      1     0     1       0       4          1          1
## # ... with 5 more variables: temp <dbl>, atemp <dbl>, hum <dbl>,
## #   windspeed <dbl>, cnt <dbl>
```

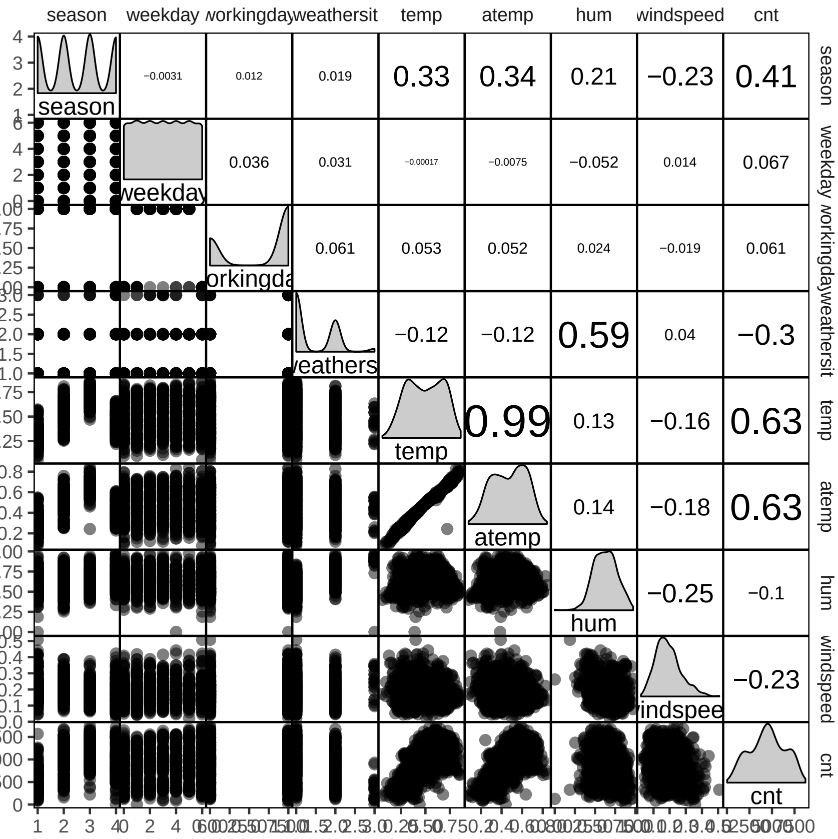
```
dim(day)
```

```
## [1] 731  14
```

```
#Mapping the amount of cnt based on 4 seasons, coloring points by weather conditions
ggplot(day,aes(cnt,season,color=weathersit)) + geom_point()
```



```
library(corrmorant)
ggcorrm(data = day[,c(3, 7:14)]) +
  lotri(geom_point(alpha = 0.5)) +
  utri_corrtext() +
  dia_names(y_pos = 0.15, size = 3) +
  dia_density(lower = 0.3, fill = "grey80", color = 1)
```



Based on the correlation plot, the upper triangle shows the correlation strength that temp(0.63), atemp(0.63) and Season(0.41) have strong correlation with the response variable(cnt). At the same time, the lower triangle scatter plot tells that when weathersit equal to 3 there might be some correlations there can be analysed.

### Lasso Regression

### Variable Selection Methods

**Splines to Capture Nonlinear Effects** Splines capture nonlinear effects by allowing the model to fit a smooth linear curve from a set of cubic functions onto the data. The general form for a cubic spline in a simple linear regression model is:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \sum_{k=1}^{K} \left( \beta_{k+3}(x_i - t_k)^3 \cdot I_{x_i > t_k} \right) + \varepsilon_i$$

Where the summation represents as series of $K$ terms that capture nonlinear effects at different intervals of the predictor variable.

### Model Selection

### Ridge Regression

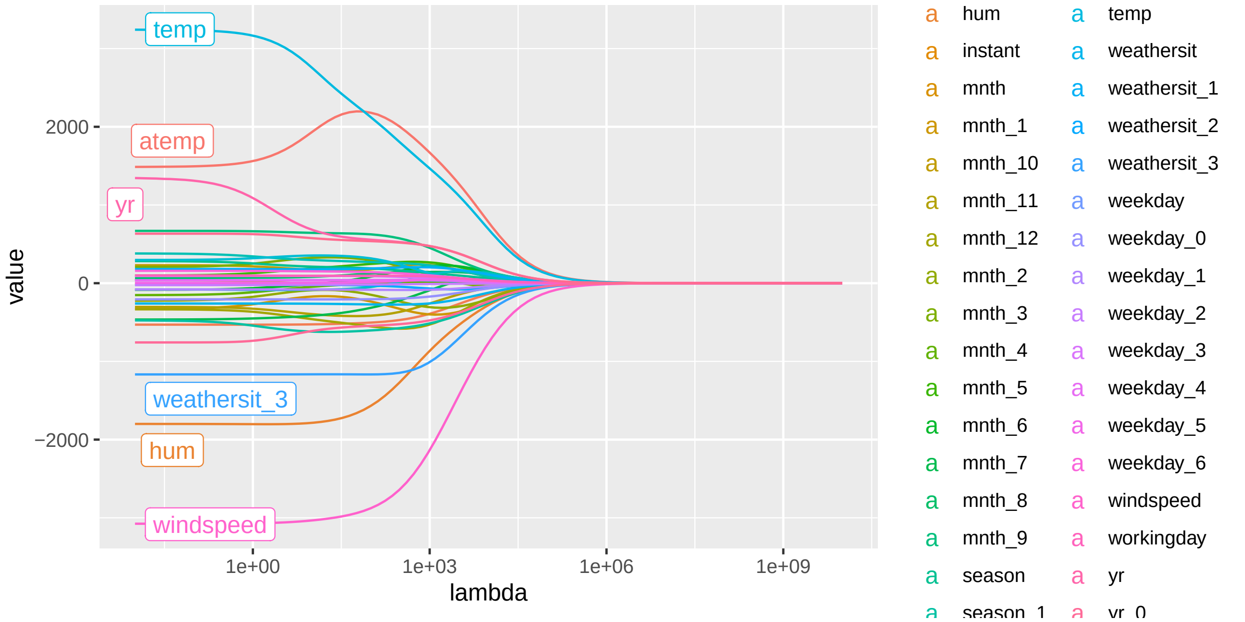
The second method that will be utilized for prediction is Ridge Regression. This method is recommneded for data with visible multi-colinearity as it seeks to minimize the following equation:

$$\sum_i (Y_i - X_{i1}\beta_1 - ... - X_{ip}\beta_p)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

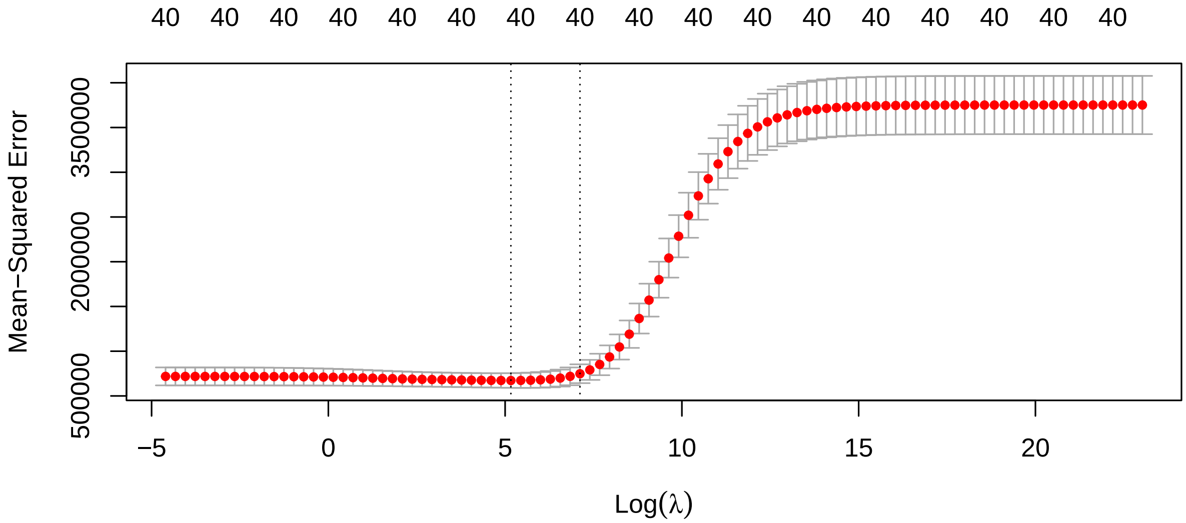
Solving this equation minimizes residual sum of squares while preventing the $\beta$'s from becoming too large. The amount of 'shrinkage' applied to the $\beta$'s is controlled by the hyperparameter $\lambda$. A small value of $\lambda$ will virtually allow the coefficients to grow very large. Inversely, A large value of $\lambda$ will reduce the coefficients closer to zero.

**Hyperparameter Tuning:** We will tune the hyperparameter, $\lambda$ using 5-fold cross-validation. The predictors we will include in training are all scaled continuous variables and all indicator variables originally created. The response variable will be the count of bike rentals. We will test lambda values between $10^{-2}$ and $10^{10}$.

Below, we see a plot showing how the coefficients of the regression models transform as we change the value of lambda. Initially, `temp` and `atemp`, two variables that have been identified as being highly correlated, have wildly differing coefficients. However, as we increase our hyperparameter, these two coefficients grow closer to each other, fixing our issue of multicollinearity. Similarly, `hum` (humidity) and `weathersit_3` (Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds) are correlated but show varying regression coefficients for low values af lambda. But, as we increase lambda, the coefficients of these two correlated variables become much more similar.



Next we will examine the plot of lambda values vs MSE as seen below. The left-most dashed vertical line represents the minimum MSE value. This value is often considered the 'best' value of lambda but often times leads to overfitting. Therefore, we will select our 'optimal' lambda as the lambda value within 1 standard error of the minimum MSE lambda to reduce variance in our predictions. This value is displayed on the plot as the right-most dashed vertical line.



The optimal value of $\lambda$ is 1232.847

### Splines to Capture Nonlinear Effects

```r
y_train <- test_set$cnt
X_test <- as.matrix(test_set %>% select(-c('cnt', 'dteday')))
```

### Model Selection

### Principal Components Analysis

The third method we utilize for prediction is Principal Components Regression. PCR is an unsupervised dimension reduction technique which seeks to draw the maximum variation from each candidate variable into individual components. In this model, we will attempt to achieve minimum test error by selecting a subset of the data in the form of the first $k$ components.

As with the other models, we will attempt to utilize cubic splines of the individual components from the training set and use 5 fold cross validation to do model selection. The principal components will only be done on continuous variables in the dataset, which include:

- Temperature
- Ambient Temperature
- Humidity
- Wind Speed

Seasonal categorical variables will be added to the regression analysis along with splines of the principal components selected for inclusion. Certain categorical variables are coded numerically (such as day of the week and month of the year) so these variables will be modeled with natural cubic splines. Other variables will be coded as simple dummy variables in the final regression model.

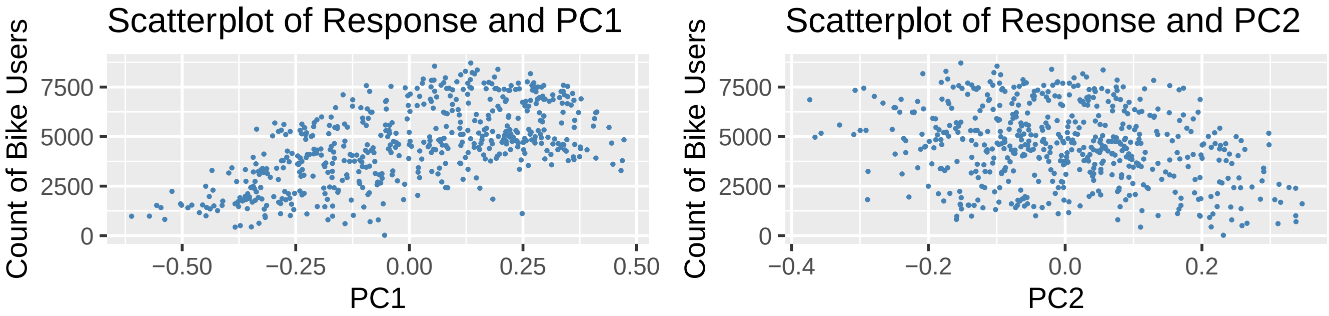
### PCA and Component Selection  Move table and plots to appendix!!

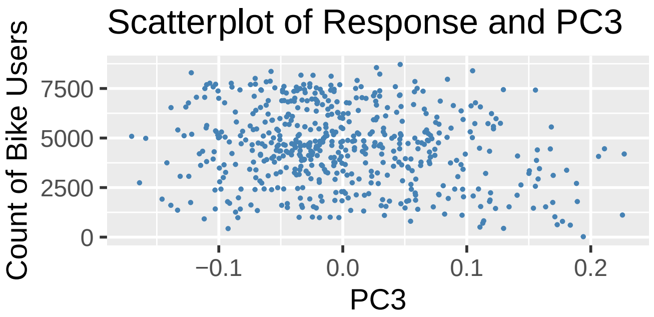
The first step is to perform PCA on the four continuous variables. Each variable is both scaled and centered in preparation for entering the principal components procedure. This ensures that each variable has comparable variability with the others.

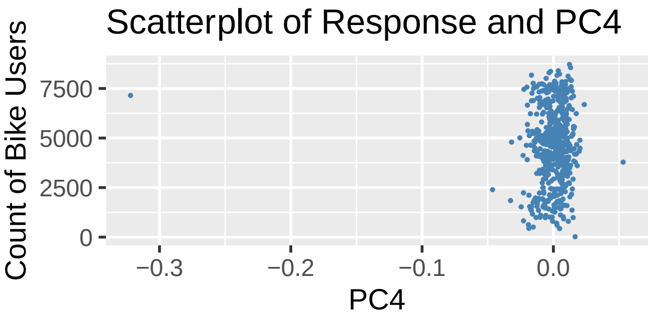
Since each variable is standardized, the total variation in the predictors is simple the number of predictors, which is 4. The following table displays the proportion of the variance from each of the four principal components as well as the cumulative variance.

As can be seen in the table, 80% of the variation in the predictors is captured in the first two components and nearly all of the variation is captured in the first three. This is partially a result of the very strong correlation between temperature and ambient temperature. From this we can conclude that principal components will be effective at reducing the dimensionality of the data.

Although the dimensionality of the data is successfully reduced, the relationship between the components and the response variable determine the overall quality of the model. The following scatterplots show the relationship between each component and the response in the training data:







The relationships between the individual components and the response are most pronounced in PC1 and PC2. PC1 looks like it may have a nonlinear effect on the response being somewhat flat on the low and high parts on the component and a positive relationship in between. The second component has what appears to be a negative effect on the response. The other two components don't appear to have any obvious relationship with Bike Sharing Counts based on the graph.

**Splines to Capture Nonlinear Effects of the Components** As in the preceding sections, cubic splines will be used to capture nonlinear effects of the components. In the case of PCA, the interpretation of the regression estimates from these splines is complicated by the fact that components are functions of a series of variables and not the original variables themselves. As such, there will be no attempt here to interpret coefficients, but rather to assess fit and predictive power.

The splines being used are natural cubic splines. The cumulative variation has suggested that PC4 contains very little of the variation from the predictors and will be omitted from the model selection procedure. The graphical analysis has suggested that while PC1 certainly seems to have nonlinear effects present, it is uncertain whether PC2 or PC3 have nonlinear effects, or whether they should be included in the model. As such, they will both be tested for spline effects.

**Model Selection** The final model is a linear regression model which utilizes cubic splines for the components PC1 - PC3 as well as day of week and month of year. Other variables consist of seasonal predictors coded as dummy variables, including:

- Weather situation
- Holiday
- Year

Working day, while present in the dataset, is excluded since it is a function of the day of the week.

Model selection was performed using 5-fold cross-validation to determine to optimal degrees of freedom parameter to use for the natural cubic spline for each component. The degrees of freedom that produced the smallest cross validation MSE for each variable are included in the final model selected.

The regression analysis was tested for including three combinations of the principal components including

- Just PC1
- PC1 and PC2
- PC1, PC2 and PC3

All other variables are allowed into the model specification.

## Conclusions

### Performance on the Holdout Dataset

### Commentary on Model Performance

## Appendix

## References

```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE)
library(tidyverse)
library(caret)
library(rsample)
library(kableExtra)
library(splines)
library(cvTools)
library(glmnet)
library(fastDummies)
library(ggrepel)
library(Metrics)
# read in data
day <- read_csv("day.csv")
day <- day %>% select(-registered, -casual)

# scale continuous variables
day_scaled <- day %>% select(temp, atemp, hum, windspeed)
day_scaled <- scale(day_scaled, center = TRUE, scale = TRUE)

# create dummy variables for indicator variables
dummyCols <- c('season', 'yr', 'mnth', 'weekday', 'weathersit')
day_scaled <- dummy_cols(day, select_columns = dummyCols)
#day_scaled <- day_scaled %>% select(-dummyCols) # remove original indicator vars

# test train split
set.seed(54321)
index <- initial_split(day_scaled, prop = 0.75)
train_set <- training(index)
test_set <- testing(index)

#Required: install remotes package
#install.packages("remotes")
# install corrmorant from the github repository
#remotes::install_github("r-link/corrmorant")

#Check out the dataset
head(day)
dim(day)
#Mapping the amount of cnt based on 4 seasons, coloring points by weather conditions
ggplot(day,aes(cnt,season,color=weathersit)) + geom_point()

library(corrmorant)
```

```r
ggcorrm(data = day[,c(3, 7:14)]) +
  lotri(geom_point(alpha = 0.5)) +
  utri_corrtext() +
  dia_names(y_pos = 0.15, size = 3) +
  dia_density(lower = 0.3, fill = "grey80", color = 1)
# create datastructures for tuning process
y_train <- train_set$cnt
X_train <- as.matrix(train_set %>% select(-c('cnt', 'dteday')))

# create grid of potential hyperparam values, train model for each value using 5-fold cv
grid <- 10^seq(-2, 10, length = 100)
cv_out <- cv.glmnet(x = X_train, y = y_train,
                    type.measure='mse',
                    nfolds = 5,
                    alpha = 0,
                    lambda = grid)

# create plot to show how beta's are affected by tuning param
## retreived from https://stackoverflow.com/questions/36656752/plotting-cv-glmnet-in-r
betas = as.matrix(cv_out$glmnet.fit$beta)
lambdas = cv_out$lambda
names(lambdas) = colnames(betas)
coefPlot <- as.data.frame(betas) %>%
              tibble::rownames_to_column("variable") %>%
              pivot_longer(-variable) %>%
              mutate(lambda=lambdas[name]) %>%
              ggplot(aes(x=lambda,y=value,col=variable)) +
              geom_line() +
              geom_label_repel(data=~subset(.x,lambda==min(lambda)),
              aes(label=variable),nudge_x=-0.5) +
              scale_x_log10()
coefPlot
# display the output of 5-fold cv for ridge regression comparing MSE to lambdas
lambdaPlot <- plot(cv_out)
lambdaPlot
y_train <- test_set$cnt
X_test <- as.matrix(test_set %>% select(-c('cnt', 'dteday')))
Xstd <- train_set %>% select(temp, atemp, hum, windspeed)
#Xstd <- scale(X, center = TRUE, scale = TRUE)

pc_out <- prcomp(Xstd)
names(pc_out)
Z <- pc_out$x

PC_var <-c(var(Z[,1]) / 4, var(Z[,2]) / 4, var(Z[,3]) / 4, var(Z[,4]) / 4)
PC_table <- data.frame(Component = c(1:4), Variance = PC_var)
PC_table <- PC_table %>% mutate(Cum_Variance = cumsum(Variance))

kable(PC_table, caption = "Proportion of Total Variance of Individual Components", digits = 3) %>% kabl

pca_train <- cbind(train_set, data.frame(Z)) %>% select(-temp, -atemp, -hum, -windspeed)

ggplot(data = pca_train, aes(x = PC1, y = cnt)) + geom_point(color = "steelblue", size = 0.25) +
  labs(title = "Scatterplot of Response and PC1", y = "Count of Bike Users", x = "PC1")

ggplot(data = pca_train, aes(x = PC2, y = cnt)) + geom_point(color = "steelblue", size = 0.25) +
  labs(title = "Scatterplot of Response and PC2", y = "Count of Bike Users", x = "PC2")

ggplot(data = pca_train, aes(x = PC3, y = cnt)) + geom_point(color = "steelblue", size = 0.25) +
  labs(title = "Scatterplot of Response and PC3", y = "Count of Bike Users", x = "PC3")

ggplot(data = pca_train, aes(x = PC4, y = cnt)) + geom_point(color = "steelblue", size = 0.25) +
  labs(title = "Scatterplot of Response and PC4", y = "Count of Bike Users", x = "PC4")

PC1_df <- c(rep(2, 625), rep(4, 625), rep(6, 625), rep(8, 625), rep(10, 625))
PC2_df <- c(rep(2, 125), rep(4, 125), rep(6, 125), rep(8, 125), rep(10, 125))
PC3_df <- c(rep(2,  25), rep(4,  25), rep(6,  25), rep(8,  25), rep(10,  25))
DOW_df <- c(rep(1,   5), rep(2,   5), rep(3,   5), rep(4,   5), rep(5 ,   5))
Mth_df <- c(1:5)
RMSE0 <- rep(0, 3125)

Grid1 <- data.frame(PC1 = PC1_df, PC2 = rep(PC2_df, 5), PC3 = rep(PC3_df, 25),
                    DOW = rep(DOW_df, 125), Month = rep(Mth_df, 625), RMSE = RMSE0)

k = 5
folds <- cvFolds(NROW(pca_train), K=k)

for(j in 1:nrow(Grid1)){
  temp <- pca_train
  temp$holdoutpred <- rep(0, nrow(temp))
  for(i in 1:k){
    train <- temp[folds$subsets[folds$which != i], ]
    val   <- temp[folds$subsets[folds$which == i], ]

    pca_ns <- lm(cnt ~ ns(PC1, df = Grid1[j, 1]) +
                       ns(PC2, df = Grid1[j, 2]) +
                       ns(PC3, df = Grid1[j, 3]) +
                       ns(weekday, df = Grid1[j, 4]) +
                       ns(mnth, df = Grid1[j, 5]) +
                       weathersit +
                       yr +
                       season +
                       holiday
                   , data = train)
    newpred <- predict(pca_ns, newdata=val)

    temp[folds$subsets[folds$which == i], ]$holdoutpred <- newpred
  }

  RMSE <- sqrt(mean((temp$holdoutpred - temp$cnt)^2))

  Grid1[j, 6] <- RMSE
}

df_vec1 <- Grid1 %>% filter(RMSE == min(Grid1$RMSE))
```

```
PC1_df <- c(rep(2, 125), rep(4, 125), rep(6, 125), rep(8, 125), rep(10, 125))
PC2_df <- c(rep(2,  25), rep(4,  25), rep(6,  25), rep(8,  25), rep(10,  25))
DOW_df <- c(rep(1,   5), rep(2,   5), rep(3,   5), rep(4,   5), rep(5 ,   5))
Mth_df <- c(1:5)
RMSE0 <- rep(0, 625)

Grid2 <- data.frame(PC1 = PC1_df, PC2 = rep(PC2_df, 5), DOW = rep(DOW_df, 25),
                    Month = rep(Mth_df, 125), RMSE = RMSE0)

k = 5
folds <- cvFolds(NROW(pca_train), K=k)

for(j in 1:nrow(Grid2)){
  temp <- pca_train
  temp$holdoutpred <- rep(0, nrow(temp))
  for(i in 1:k){
    train <- temp[folds$subsets[folds$which != i], ]
    val   <- temp[folds$subsets[folds$which == i], ]

    pca_ns <- lm(cnt ~ ns(PC1, df = Grid2[j, 1]) +
                   ns(PC2, df = Grid2[j, 2]) +
                   ns(weekday, df = Grid2[j, 3]) +
                   ns(mnth, df = Grid2[j, 4]) +
                   weathersit +
                   yr +
                   season +
                   holiday
                 , data = train)
    newpred <- predict(pca_ns, newdata=val)

    temp[folds$subsets[folds$which == i], ]$holdoutpred <- newpred
  }

  RMSE <- sqrt(mean((temp$holdoutpred - temp$cnt)^2))

  Grid2[j, 5] <- RMSE
}

df_vec2 <- Grid2 %>% filter(RMSE == min(Grid2$RMSE))


PC1_df <- c(rep(2,  25), rep(4,  25), rep(6,  25), rep(8,  25), rep(10,  25))
DOW_df <- c(rep(1,   5), rep(2,   5), rep(3,   5), rep(4,   5), rep(5 ,   5))
Mth_df <- c(1:5)
RMSE0 <- rep(0, 125)

Grid3 <- data.frame(PC1 = PC1_df, DOW = rep(DOW_df, 5),  Month = rep(Mth_df, 25), RMSE = RMSE0)

k = 5
folds <- cvFolds(NROW(pca_train), K=k)

for(j in 1:nrow(Grid3)){
  temp <- pca_train
  temp$holdoutpred <- rep(0, nrow(temp))
  for(i in 1:k){
    train <- temp[folds$subsets[folds$which != i], ]
    val   <- temp[folds$subsets[folds$which == i], ]

    pca_ns <- lm(cnt ~ ns(PC1, df = Grid3[j, 1]) +
                   ns(weekday, df = Grid3[j, 2]) +
                   ns(mnth, df = Grid3[j, 3]) +
                   weathersit +
                   yr +
                   season +
                   holiday
                 , data = train)
    newpred <- predict(pca_ns, newdata=val)

    temp[folds$subsets[folds$which == i], ]$holdoutpred <- newpred
  }

  RMSE <- sqrt(mean((temp$holdoutpred - temp$cnt)^2))

  Grid3[j, 4] <- RMSE
}

df_vec3 <- Grid3 %>% filter(RMSE == min(Grid3$RMSE))
```