ST 563 Final Project - Bike Sharing Data

Team #1

Peter Lung, David Shaw, Yijia Cai

12/2/2021

Contents

Introduction . . . 2
Methods . . . 3
Exploratory Data Analysis . . . 3
Daily Bike Rental Count . . . 3
Weather and Season Multi-collinearity . . . 3
Correlation Matrix . . . 3
Splines for Non-Linear Predictors . . . 5
Ridge Regression . . . 6
Hyperparameter Tuning: . . . 6
Splines to Capture Nonlinear Effects: . . . 7
Model Selection . . . 7
Lasso Regression . . . 9
Hyperparameter Tuning: . . . 9
Splines to Capture Nonlinear Effects: . . . 9
Model Selection . . . 11
Principal Components Analysis . . . 12
PCA and Component Selection . . . 12
Splines to Capture Nonlinear Effects of the Components . . . 13
Model Selection . . . 13
Conclusions . . . 15
Performance on the Holdout Dataset . . . 15
Commentary on Model Performance . . . 15
Final Thoughts . . . 15
Appendix . . . 16
References . . . 23

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

Daily Bike Rental Count

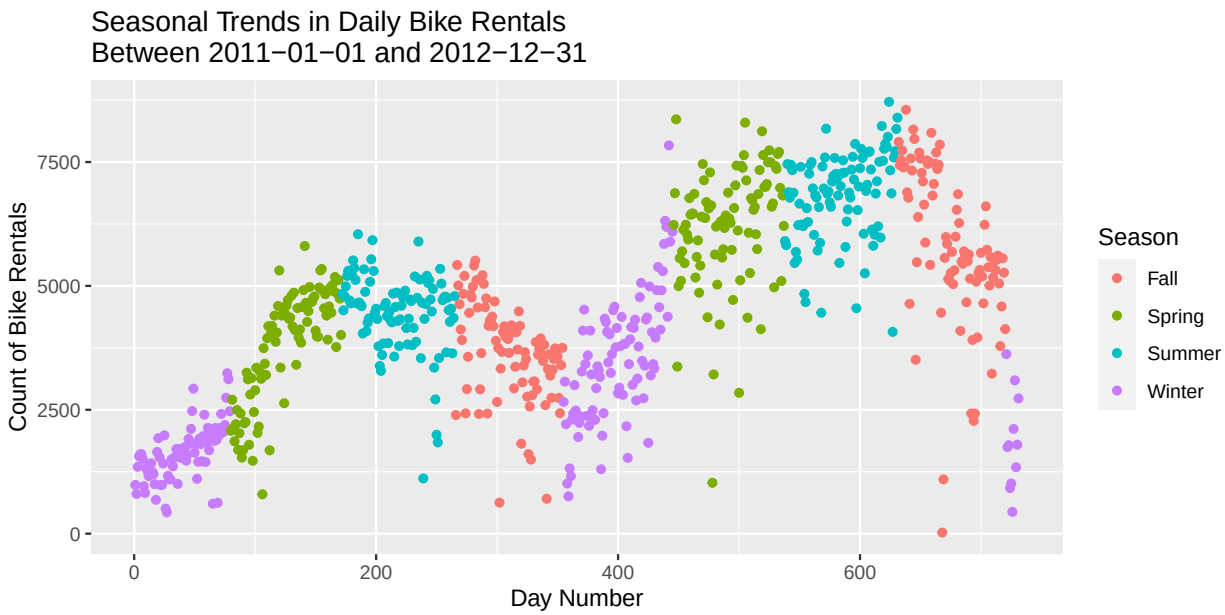


Figure 1: Daily Bike Rentals

In figure 1, we plot the day of year vs the count of bike rentals. We notice a seasonal trend: bike rentals increase starting at the beginning of spring, reach a maximum in the summer, and then start to decrease in the fall. This trend is not surprising as it is expected that bike rentals will be more popular in the warmer seasons.

Weather and Season Multi-collinearity

In figure 2, we notice the summers have more clear days and less mist/light precipitation days. This leads us to believe that season and precipitation may exhibit multi-collinearity with the daily bike rental count.

Correlation Matrix

Based on the correlation plot in figure 3, 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.

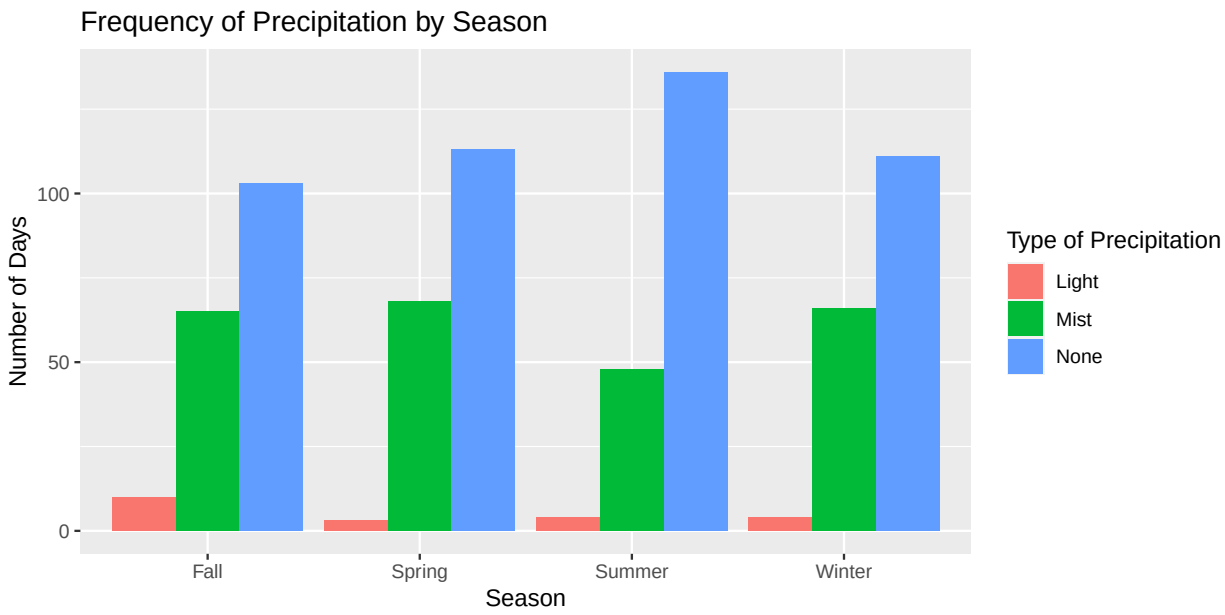


Figure 2: Weather trends

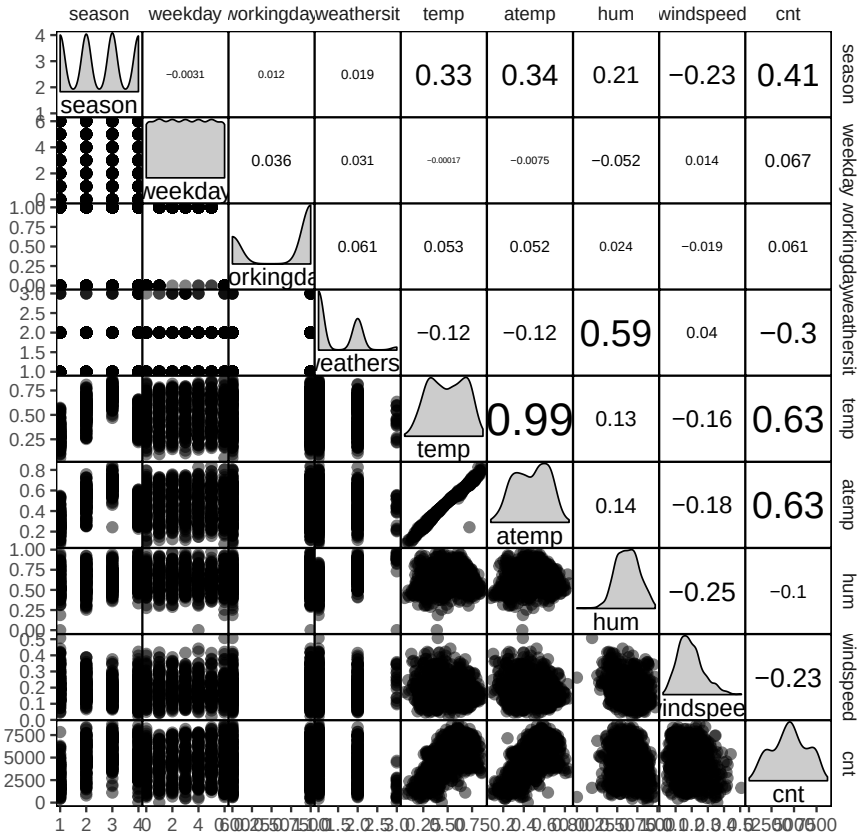


Figure 3: Correlation Matrix

Splines for Non-Linear Predictors

Some predictors exhibit non-linear relationships with the response variable. 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.

The following plots in figure 4 show two variables, Humidity and Windspeed, which have nonlinear relationships with Count and will be modeled with splines.

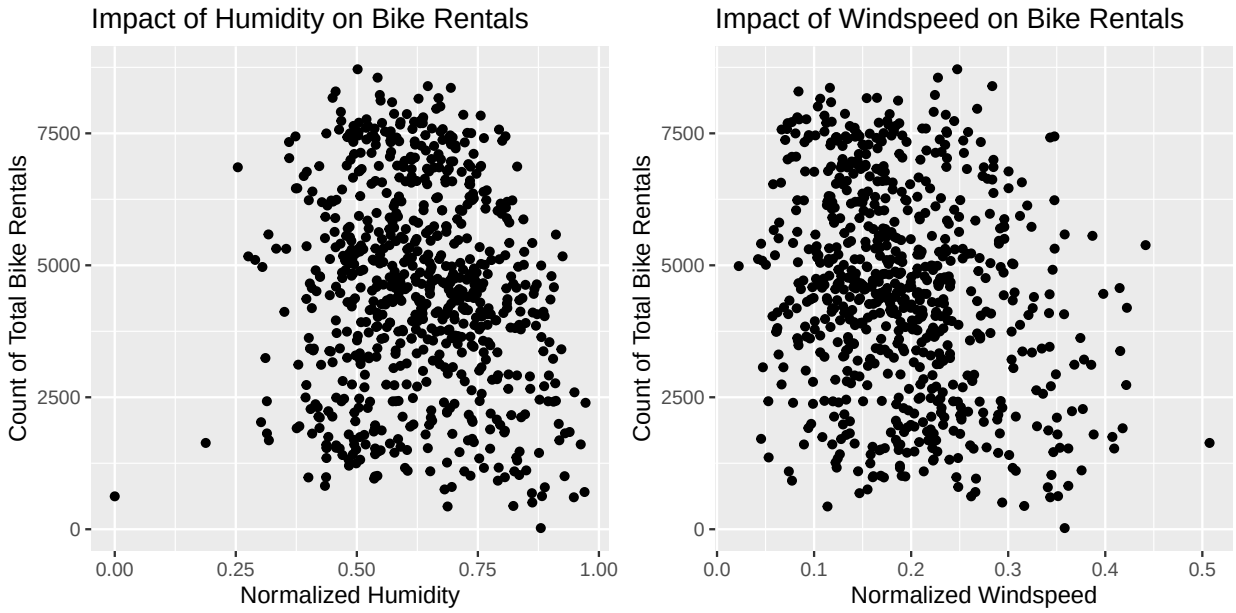


Figure 4: Non-linear Predictors

Generating splines for these data may boost some of the predictive power of these variables and could lead to stronger performance of models.

Ridge Regression

The first method that will be utilized for prediction is Ridge Regression. This method is recommeneded 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 β's from becoming too large. The amount of 'shrinkage' applied to the β's is controlled by the hyperparameter λ. A small value of λ will virtually allow the coefficients to grow very large. Inversely, A large value of λ will reduce the coefficients closer to zero.

Hyperparameter Tuning: We will tune the hyperparameter, λ 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 in figure 5, 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.

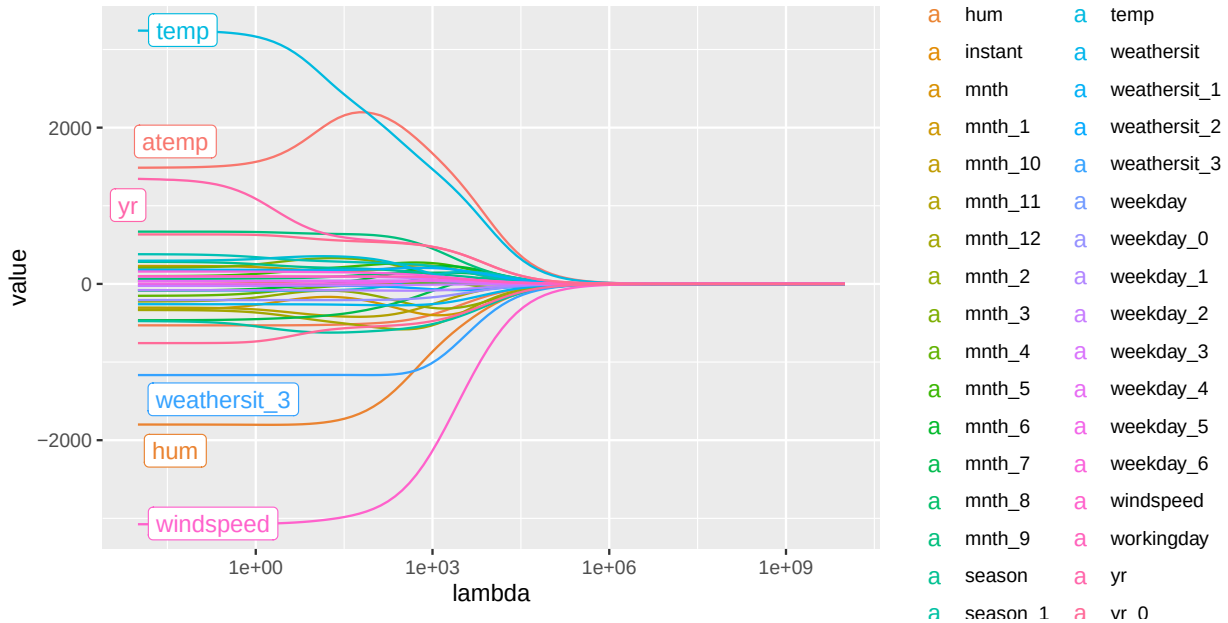


Figure 5: Ridge Hyperparameter Tuning, Impact of Lambda on Coefficients

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.

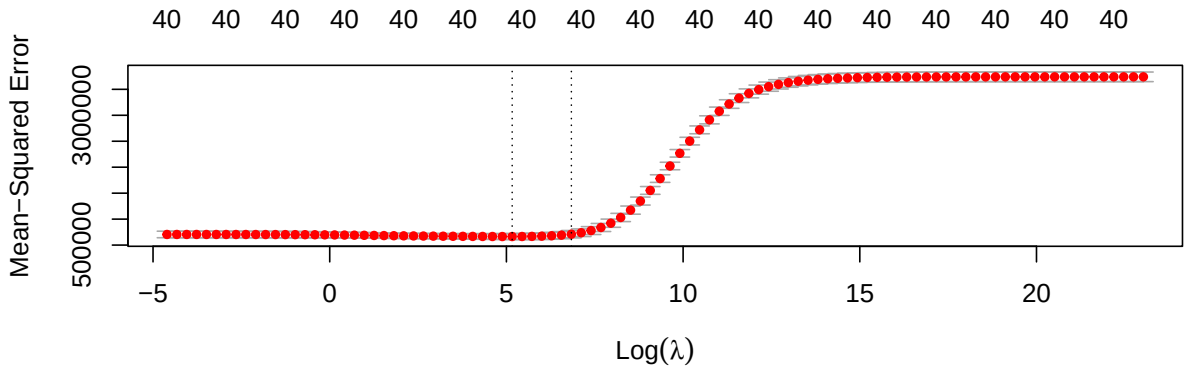


Figure 6: Ridge Hyperparameter Tuning, Choosing Optimal Lambda

The optimal value of λ is 932.603

Splines to Capture Nonlinear Effects: Next, we will fit a ridge regression model on the same training dataset, but, with humidity and windspeed variables transformed with splines. This transformation will attempt to boost the predictive power of humidity and windspeed.

Similar to above, we will train the model using 5-fold cross-validation. The optimal λ will be selected using the 1 standard-error rule. This value is displayed on the in figure 7 below as the right-most dashed vertical line.

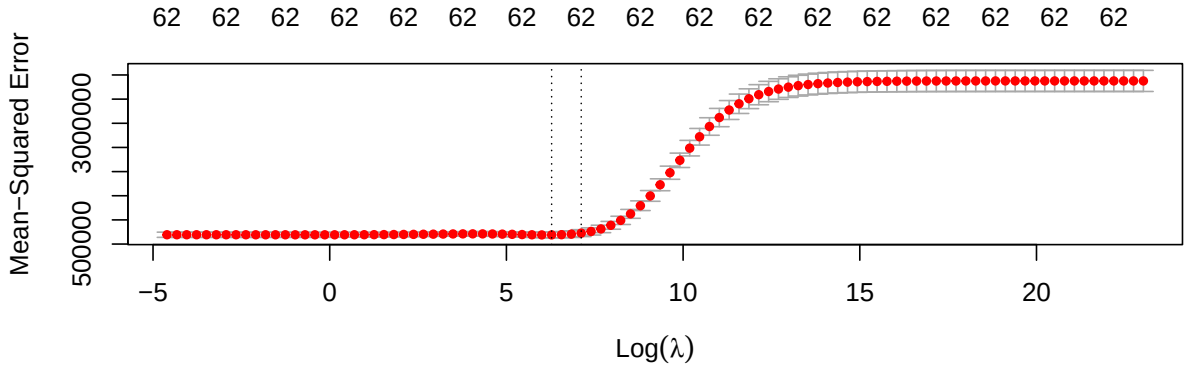


Figure 7: Ridge+Spline Hyperparameter Tuning, Choosing Optimal Lambda

The optimal value of λ is 1232.847

Model Selection Finally, we will compare the two models, ridge regression and ridge regression + piecewise spline, against our test dataset. The model with lower test MSE will be determined as better.

Table 1: Results of Ridge Regression Models on Test Dataset

	ridgeReg	ridgeSplineReg
MSE	571886.3	775098.7
Squared-Bias	3770262.9	3358518.2
Variance	2486937.1	2230095.4

Ridge regression without splines has a lower MSE. Therefore, we will use this model for final comparison.

Lasso Regression

Lasso as a shrinkage method uses a penalty term involving sum of the absolute values of the regression coefficients.

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

Lasso estimates of β_j by minimizing residual sum of squares for $\lambda >= 0$. Additionally, lasso regression has the advantage of eliminating terms from the model (setting coefficients to zero) which classifies the technique as a dimension reduction method.

Hyperparameter Tuning: Similar to the Ridge regression, we will see the coefficients of the predictors change as lambda increases of based on the figure below. The solution path selected variables: temp, yr, atemp, mnth_9, weathersit_3, season_1, hum, and windspeed as the most influential variables in the model.

Although atemp and temp are highly correlated with each other (as seen in the correlation matrix above) and might be a source of multicollinearity, the lasso method alleviates this problem be forcing the coefficients of correlated predictors closer together.

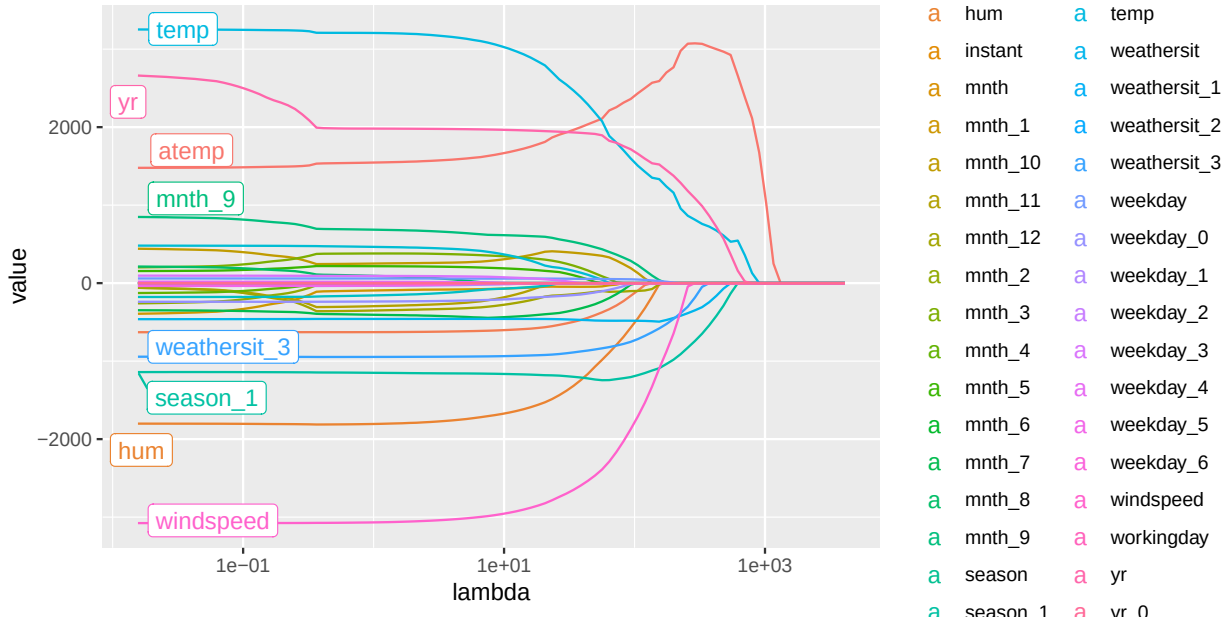


Figure 8: Lasso Hyperparameter Tuning, Impact of Lambda on Coefficients

The lambda values with minimum CV error and chosen by the one standard rule are shown below in figure 9, along with the corresponding coefficient estimates.

The optimal value of λ is 64

Splines to Capture Nonlinear Effects: Now, we will attempt to fit a lasso regression model on the training datset where a piecewise spline transformation is applied to humidity and windspeed. This model will attempt to alleviate the non-linearity concerns spurred by these two predictors. Again the optimal value will be chosen using the one standard-error rule, as seen in figure 10.

The optimal value of λ with splines is 86.975

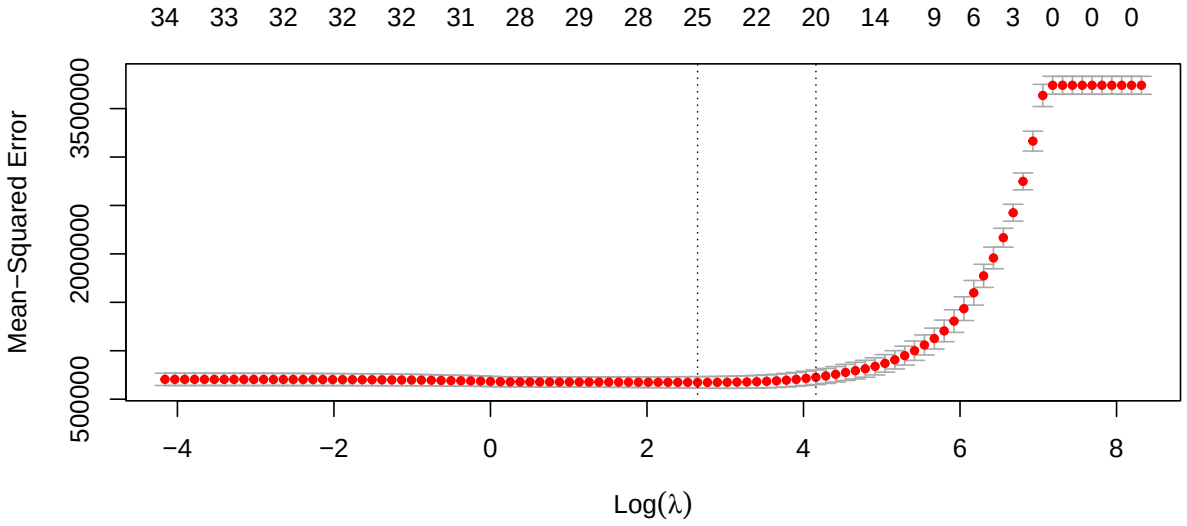


Figure 9: Lasso Hyperparameter Tuning, Choosing Optimal Lambda

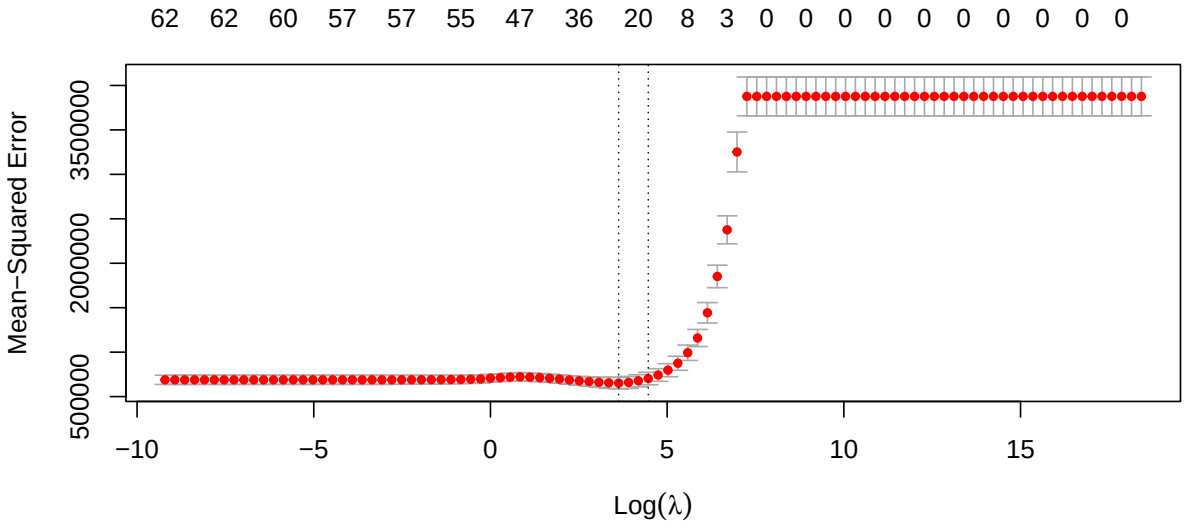


Figure 10: Lasso+Spline Hyperparameter Tuning, Choosing Optimal Lambda

Model Selection We will now compare the two models, lasso regression and lasso regression + splines, against our test dataset. The model with lower test MSE will be determined as better.

Table 2: Results of Lasso Regression Models on Test Dataset

	lassoReg	lassoSplineReg
MSE	554607.2	792037
Squared-Bias	3769862.6	3356247
Variance	2759616.3	2546323

Lasso regression without splines has a lower MSE, so we choose it as our final model.

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 as there is a natural order to their numbering. This will capture effects such as the increase in counts during the summer as compared to the winter. Season will be modeled with a quadratic functional form and weather situation will be modeled linearly. Other variables, such as holidays and year, will be coded as simple dummy variables in the final regression model.

PCA and Component Selection 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.

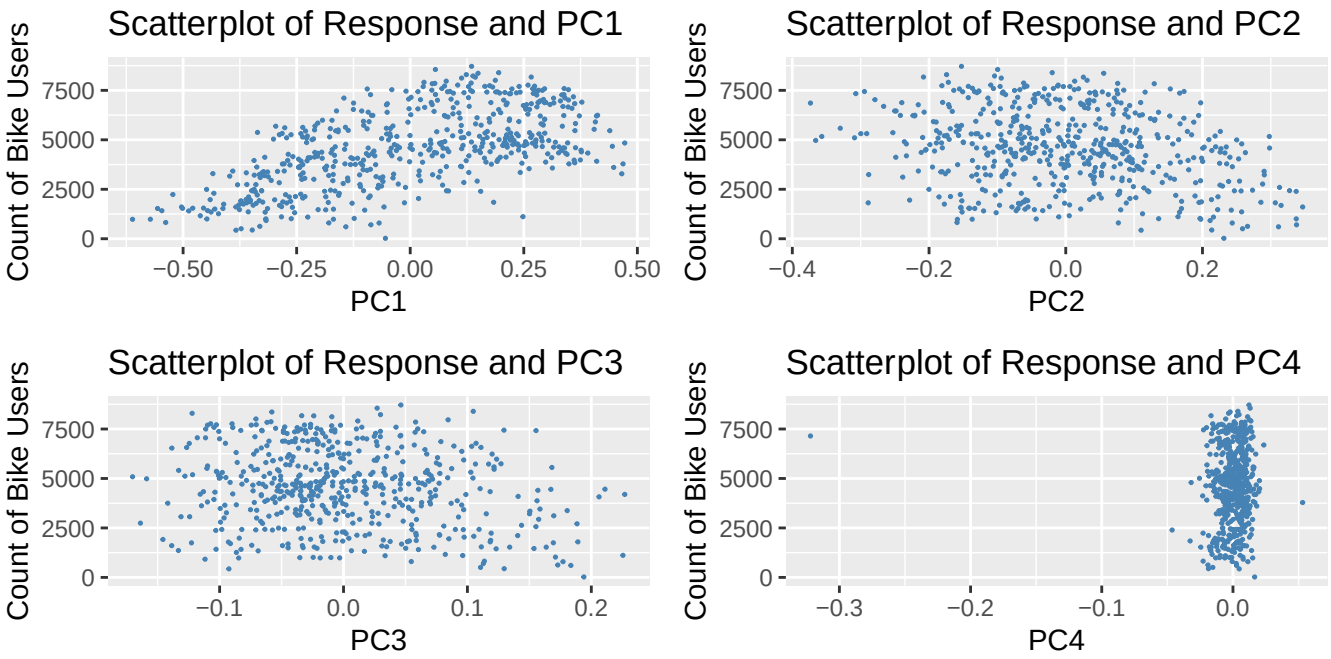
```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
```

Table 3: Proportion of Total Variance of Individual Components

Component	Variance	Cum_Variance
1	0.015	0.015
2	0.005	0.020
3	0.001	0.021
4	0.000	0.021

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 considered 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.

The idea of modeling this way is to reduce the variable dimensions, but model the nonlinear effects with natural splines such that the bias is minimized.

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

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

All other variables are allowed into the model specification. The results from the cross validation for each of those model specifications is displayed in the table.

Table 4: Best Model for Each Specification (Natural Spline DF)

Model	PC1	PC2	PC3	DOW	Month	RMSE
PC1, PC2 & PC3	3	4	4	1	1	695.0
PC1 & PC2	6	2	0	1	1	745.8
PC1 only	4	0	0	2	2	771.5

Based on these results, the final PCA model includes the first three components with the following natural spline degrees of freedom:

- PC1: **3**
- PC2: **4**
- PC3: **4**
- Day of Week: **1**
- Month of Year: **1**

Regression output from the final model is displayed below.

term	estimate	std.error	statistic	p.value
(Intercept)	2634.518	430.793	6.116	0.000
ns(PC1, df = PC1_df)1	4093.674	181.643	22.537	0.000
ns(PC1, df = PC1_df)2	3319.825	527.025	6.299	0.000
ns(PC1, df = PC1_df)3	849.618	202.896	4.187	0.000
ns(PC2, df = PC2_df)1	-76.099	244.062	-0.312	0.755
ns(PC2, df = PC2_df)2	-702.020	207.155	-3.389	0.001
ns(PC2, df = PC2_df)3	-1473.856	569.965	-2.586	0.010
ns(PC2, df = PC2_df)4	-1391.283	253.861	-5.480	0.000
ns(PC3, df = PC3_df)1	-329.787	234.414	-1.407	0.160
ns(PC3, df = PC3_df)2	-26.387	203.394	-0.130	0.897
ns(PC3, df = PC3_df)3	-1267.306	560.731	-2.260	0.024
ns(PC3, df = PC3_df)4	-2086.228	245.887	-8.485	0.000
ns(weekday, df = DOW_df)	603.954	110.967	5.443	0.000
ns(mnth, df = Mth_df)	-569.115	221.920	-2.565	0.011
weathersit	-496.566	76.825	-6.464	0.000
yr	2016.172	60.299	33.436	0.000
poly(season, 2)1	12732.707	1320.650	9.641	0.000
poly(season, 2)2	-3629.636	1082.006	-3.355	0.001
holiday	-563.763	171.477	-3.288	0.001

Conclusions

Performance on the Holdout Dataset

Below in table 5, we show a comparison of the model performance on test set:

Table 5: Results of Models on Test Dataset

	Ridge	Ridge+Spline	Lasso	Lasso+Spline	PCA
RMSE	756.232	880.397	744.72	889.965	644.67

The best performing model is PCA. Its RMSE on the holdout dataset is **644.67**.

Commentary on Model Performance

In the PCA model, the data reduction achieved was minimal, as three out of four principal components were present in the final model (the fourth PC not being considered for inclusion to enable at least some dimension reduction). The splines were able to model nonlinear effects in the components, however, and the holdout RMSE demonstrates a good test error for this model.

Ridge and Lasso regression methods fell short of modeling non-linear variables. Additionally, when splines were incorporated into ridge/lasso, model performance grew worse.

Final Thoughts

Moving forward, collecting additional data would be beneficial to the performance of lasso/ridge/pca techniques. The original dataset contains 14 variables (including the response variable). Of which, 7 variables are used to describe the day of the year and 1 variable represents the row number in the dataset. In reality, this dataset provides 6 uniquely collected variables to predict the daily count of bike renters. Looking back, perhaps due to the limited number of variables to start with, shrinkage or dimension reduction methods may not have been appropriate.

Another idea to consider is the potential to fit a time-series model rather than a learning model. The dataset exhibits strong seasonal trends as well as an overall yearly increase in daily bike rentals. Given a longer range of data, perhaps the time-series model would outperform a learning model.

Appendix

```
library(tidyverse)
library(caret)
library(rsample)
library(kableExtra)
library(splines)
library(cvTools)
library(glmnet)
library(fastDummies)
library(ggrepel)
library(Metrics)
library(gridExtra)
library(splines)

# 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)

#### Create dataset with spline modified vars ####
# create splines for humidity
kn <- quantile(day_scaled$hum, probs = seq(.10, .90, by = .1)) # Define region boundaries
basis <- bs(day_scaled$hum,
                degree = 3,
                knots = kn) # Cubic splines basis functions
df <- data.frame(basis)
df <- df %>% rename_with( ~ paste0("hum_", .x))
day_scaled_spline <- merge(day_scaled, df, by.x=0, by.y=0)
day_scaled_spline <- day_scaled_spline %>% select(-c('Row.names', 'hum'))

# create splines for windspeed
kn <- quantile(day_scaled$windspeed, probs = seq(.10, .90, by = .1)) # Define region boundaries
basis <- bs(day_scaled$windspeed,
                degree = 3,
                knots = kn) # Cubic splines basis functions
df <- data.frame(basis)
df <- df %>% rename_with( ~ paste0("windspeed_", .x))
day_scaled_spline <- merge(day_scaled_spline, df, by.x=0, by.y=0)
```

```r
day_scaled_spline <- day_scaled_spline %>% select(-c('Row.names', 'windspeed'))

# test train split
index <- initial_split(day_scaled_spline, prop = 0.75)
train_set_spline <- training(index)
test_set_spline <- 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)
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)

humCnt <- ggplot(data=day) +
            geom_point(mapping=aes(x=hum, y=cnt)) +
            ggtitle("Impact of Humidity on Bike Rentals") +
            xlab("Normalized Humidity") + ylab("Count of Total Bike Rentals")
windCnt <- ggplot(data=day) +
            geom_point(mapping=aes(x=windspeed, y=cnt)) +
            ggtitle("Impact of Windspeed on Bike Rentals") +
            xlab("Normalized Windspeed") + ylab("Count of Total Bike Rentals")
grid.arrange(humCnt, windCnt, nrow=1)

# create datastructures for tuning process
y_train <- train_set$cnt
X_train <- as.matrix(train_set %>% select(-c('cnt', 'dteday')))
#X_train <- as.matrix(train_set[,-c(1:9)])

# create grid of potential hyperparam values, train model for each value using 5-fold cv
set.seed(54321)
grid <- 2^seq(-6, 12, length = 100)
cv_out1 <- cv.glmnet(x = X_train, y = y_train,
                     type.measure='mse',
                     nfolds = 5,
                     alpha = 1,
                     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_out1$glmnet.fit$beta)
lambdas1 = cv_out1$lambda
```

```
names(lambdas1) = colnames(betas)
coefPlot <- as.data.frame(betas) %>%
              tibble::rownames_to_column("variable") %>%
              pivot_longer(-variable) %>%
              mutate(lambda=lambdas1[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
plot(cv_out1)

# create datastructures for tuning process
y_train_spline <- train_set_spline$cnt
X_train_spline <- as.matrix(train_set_spline %>% select(-c('cnt', 'dteday')))

# create grid of potential hyperparam values, train model for each value using 5-fold cv
set.seed(54321)
grid <- 10^seq(-4, 8, length = 100)
cv_out3 <- cv.glmnet(x = X_train_spline, y = y_train_spline,
                     type.measure='mse',
                     nfolds = 5,
                     alpha = 1,
                     lambda = grid)

plot(cv_out3)

# evaluate performance of ridge regression model
y_test <- test_set$cnt
X_test <- as.matrix(test_set %>% select(-c('cnt', 'dteday')))
pred1 <- predict(cv_out1, X_test)
lassoReg <- c()
lassoReg <- append(lassoReg, mean((pred1 - y_test)^2))       # mse
lassoReg <- append(lassoReg, mean((mean(pred1) - y_test)^2)) # squared bias
lassoReg <- append(lassoReg, mean((pred1 - mean(pred1))^2))   # variance

# evaluate performance of ridge+spline regression model
y_test <- test_set_spline$cnt
X_test <- as.matrix(test_set_spline %>% select(-c('cnt', 'dteday')))
pred3 <- predict(cv_out3, X_test)
lassoSplineReg <- c()
lassoSplineReg <- append(lassoSplineReg, mean((pred3 - y_test)^2))       # mse
lassoSplineReg <- append(lassoSplineReg, mean((mean(pred3) - y_test)^2)) # squared bias
lassoSplineReg <- append(lassoSplineReg, mean((pred3 - mean(pred3))^2))   # variance

# print table
results1 <- data.frame(lassoReg, lassoSplineReg, row.names = c('MSE', 'Squared-Bias', 'Variance'))
kable(results1,
      caption = "Results of Lasso Regression Models on Test Dataset",
      digits = 3) %>%
```

```
        kable_styling(latex_options = "hold_position")

# 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
set.seed(54321)
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

plot(cv_out)

# create datastructures for tuning process
y_train_spline <- train_set_spline$cnt
X_train_spline <- as.matrix(train_set_spline %>% select(-c('cnt', 'dteday')))

# create grid of potential hyperparam values, train model for each value using 5-fold cv
set.seed(54321)
grid <- 10^seq(-2, 10, length = 100)
cv_out2 <- cv.glmnet(x = X_train_spline, y = y_train_spline,
                    type.measure='mse',
                    nfolds = 5,
                    alpha = 0,
                    lambda = grid)

plot(cv_out2)

# evaluate performance of ridge regression model
y_test <- test_set$cnt
X_test <- as.matrix(test_set %>% select(-c('cnt', 'dteday')))
pred <- predict(cv_out, X_test)
ridgeReg <- c()
```

```
ridgeReg <- append(ridgeReg, mean((pred - y_test)^2))       # mse
ridgeReg <- append(ridgeReg, mean((mean(pred) - y_test)^2)) # squared bias
ridgeReg <- append(ridgeReg, mean((pred - mean(pred))^2))   # variance

# evaluate performance of ridge+spline regression model
y_test <- test_set_spline$cnt
X_test <- as.matrix(test_set_spline %>% select(-c('cnt', 'dteday')))
pred <- predict(cv_out2, X_test)
ridgeSplineReg <- c()
ridgeSplineReg <- append(ridgeSplineReg, mean((pred - y_test)^2))       # mse
ridgeSplineReg <- append(ridgeSplineReg, mean((mean(pred) - y_test)^2)) # squared bias
ridgeSplineReg <- append(ridgeSplineReg, mean((pred - mean(pred))^2))   # variance

# print table
results <- data.frame(ridgeReg, ridgeSplineReg, row.names = c('MSE', 'Squared-Bias', 'Variance'))
kable(results,
      caption = "Results of Ridge Regression Models on Test Dataset",
      digits = 3) %>%
        kable_styling(latex_options = "hold_position")

# Run PCA on the continuous variables
Xstd <- train_set %>% select(temp, atemp, hum, windspeed)
pc_out <- prcomp(Xstd)

# Extend PCA to test set for holdout analysis
Xstd_test <- predict(pc_out, test_set)
pca_test <- cbind(test_set, Xstd_test) %>% select(-temp, -atemp, -hum, -windspeed)

# Create table to display variation
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))
```

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)
library(gridExtra)
library(splines)
# 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)

#### Create dataset with spline modified vars ####
# create splines for humidity
kn <- quantile(day_scaled$hum, probs = seq(.10, .90, by = .1)) # Define region boundaries
basis <- bs(day_scaled$hum,
                degree = 3,
                knots = kn) # Cubic splines basis functions
df <- data.frame(basis)
df <- df %>% rename_with( ~ paste0("hum_", .x))
day_scaled_spline <- merge(day_scaled, df, by.x=0, by.y=0)
day_scaled_spline <- day_scaled_spline %>% select(-c('Row.names', 'hum'))

# create splines for windspeed
```

```
kn <- quantile(day_scaled$windspeed, probs = seq(.10, .90, by = .1)) # Define region boundaries
basis <- bs(day_scaled$windspeed,
               degree = 3,
               knots = kn) # Cubic splines basis functions
df <- data.frame(basis)
df <- df %>% rename_with( ~ paste0("windspeed_", .x))
day_scaled_spline <- merge(day_scaled_spline, df, by.x=0, by.y=0)
day_scaled_spline <- day_scaled_spline %>% select(-c('Row.names', 'windspeed'))

# test train split
index <- initial_split(day_scaled_spline, prop = 0.75)
train_set_spline <- training(index)
test_set_spline <- testing(index)
# give labels to season
day$seasonTag <- ifelse(day$season==1, 'Winter',
                        ifelse(day$season==2, 'Spring',
                               ifelse(day$season==3, 'Summer', 'Fall')))

# plot seasonal trends
ggplot(data = day, aes(x = instant, y = cnt)) +
    geom_point(aes(colour = factor(seasonTag))) +
    labs(title = 'Seasonal Trends in Daily Bike Rentals\nBetween 2011-01-01 and 2012-12-31',
         x = 'Day Number',
         y = 'Count of Bike Rentals',
         color = 'Season')
# give labels to weather
day$weathersitTag <- ifelse(day$weathersit==1, 'None',
                        ifelse(day$weathersit==2, 'Mist',
                               ifelse(day$weathersit==3, 'Light', 'Heavy')))

# plot seasonal trends
ggplot(data = day, mapping = aes(x = seasonTag, fill = weathersitTag)) +
    geom_bar(position = 'dodge') +
    labs(title = 'Frequency of Precipitation by Season',
         x = 'Season',
         y = 'Number of Days',
         fill = 'Type of Precipitation')

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)
humCnt <- ggplot(data=day) +
            geom_point(mapping=aes(x=hum, y=cnt)) +
            ggtitle("Impact of Humidity on Bike Rentals") +
            xlab("Normalized Humidity") + ylab("Count of Total Bike Rentals")
windCnt <- ggplot(data=day) +
            geom_point(mapping=aes(x=windspeed, y=cnt)) +
            ggtitle("Impact of Windspeed on Bike Rentals") +
            xlab("Normalized Windspeed") + ylab("Count of Total Bike Rentals")
grid.arrange(humCnt, windCnt, nrow=1)
```

```r
# 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
set.seed(54321)
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$lambda
names(lambdas) = colnames(betas)
coefPlot <- as.data.frame(betas) %>%
              tibble::rownames_to_column("variable") %>%
              pivot_longer(-variable) %>%
              mutate(lambda=lambdas[name]) %>%
              ggplot(aes(x=lambda,y=value,col=variable)) +
              geom_line() +
              geom_label_repel(data=~subset(.x,lambda==min(lambda)),
              aes(label=variable),nudge_x=-0.5) +
              scale_x_log10()
coefPlot
# display the output of 5-fold cv for ridge regression comparing MSE to lambdas
plot(cv_out)
# create datastructures for tuning process
y_train_spline <- train_set_spline$cnt
X_train_spline <- as.matrix(train_set_spline %>% select(-c('cnt', 'dteday')))

# create grid of potential hyperparam values, train model for each value using 5-fold cv
set.seed(54321)
grid <- 10^seq(-2, 10, length = 100)
cv_out2 <- cv.glmnet(x = X_train_spline, y = y_train_spline,
                    type.measure='mse',
                    nfolds = 5,
                    alpha = 0,
                    lambda = grid)

plot(cv_out2)
# evaluate performance of ridge regression model
y_test <- test_set$cnt
X_test <- as.matrix(test_set %>% select(-c('cnt', 'dteday')))
pred <- predict(cv_out, X_test)
ridgeReg <- c()
ridgeReg <- append(ridgeReg, mean((pred - y_test)^2))       # mse
ridgeReg <- append(ridgeReg, mean((mean(pred) - y_test)^2)) # squared bias
ridgeReg <- append(ridgeReg, mean((pred - mean(pred))^2))   # variance
```

```r
# evaluate performance of ridge+spline regression model
y_test <- test_set_spline$cnt
X_test <- as.matrix(test_set_spline %>% select(-c('cnt', 'dteday')))
pred <- predict(cv_out2, X_test)
ridgeSplineReg <- c()
ridgeSplineReg <- append(ridgeSplineReg, mean((pred - y_test)^2))       # mse
ridgeSplineReg <- append(ridgeSplineReg, mean((mean(pred) - y_test)^2)) # squared bias
ridgeSplineReg <- append(ridgeSplineReg, mean((pred - mean(pred))^2))   # variance

# print table
results <- data.frame(ridgeReg, ridgeSplineReg, row.names = c('MSE', 'Squared-Bias', 'Variance'))
kable(results,
      caption = "Results of Ridge Regression Models on Test Dataset",
      digits = 3) %>%
        kable_styling(latex_options = "hold_position")
# create datastructures for tuning process
y_train <- train_set$cnt
X_train <- as.matrix(train_set %>% select(-c('cnt', 'dteday')))
#X_train <- as.matrix(train_set[,-c(1:9)])

# create grid of potential hyperparam values, train model for each value using 5-fold cv
set.seed(54321)
grid <- 2^seq(-6, 12, length = 100)
cv_out1 <- cv.glmnet(x = X_train, y = y_train,
                     type.measure='mse',
                     nfolds = 5,
                     alpha = 1,
                     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_out1$glmnet.fit$beta)
lambdas1 = cv_out1$lambda
names(lambdas1) = colnames(betas)
coefPlot <- as.data.frame(betas) %>%
              tibble::rownames_to_column("variable") %>%
              pivot_longer(-variable) %>%
              mutate(lambda=lambdas1[name]) %>%
              ggplot(aes(x=lambda,y=value,col=variable)) +
              geom_line() +
              geom_label_repel(data=~subset(.x,lambda==min(lambda)),
              aes(label=variable),nudge_x=-0.5) +
              scale_x_log10()
coefPlot
# display the output of 5-fold cv for ridge regression comparing MSE to lambdas
plot(cv_out1)
# create datastructures for tuning process
y_train_spline <- train_set_spline$cnt
X_train_spline <- as.matrix(train_set_spline %>% select(-c('cnt', 'dteday')))

# create grid of potential hyperparam values, train model for each value using 5-fold cv
set.seed(54321)
grid <- 10^seq(-4, 8, length = 100)
```

```
cv_out3 <- cv.glmnet(x = X_train_spline, y = y_train_spline,
                     type.measure='mse',
                     nfolds = 5,
                     alpha = 1,
                     lambda = grid)

plot(cv_out3)
# evaluate performance of ridge regression model
y_test <- test_set$cnt
X_test <- as.matrix(test_set %>% select(-c('cnt', 'dteday')))
pred1 <- predict(cv_out1, X_test)
lassoReg <- c()
lassoReg <- append(lassoReg, mean((pred1 - y_test)^2))       # mse
lassoReg <- append(lassoReg, mean((mean(pred1) - y_test)^2)) # squared bias
lassoReg <- append(lassoReg, mean((pred1 - mean(pred1))^2))   # variance

# evaluate performance of ridge+spline regression model
y_test <- test_set_spline$cnt
X_test <- as.matrix(test_set_spline %>% select(-c('cnt', 'dteday')))
pred3 <- predict(cv_out3, X_test)
lassoSplineReg <- c()
lassoSplineReg <- append(lassoSplineReg, mean((pred3 - y_test)^2))       # mse
lassoSplineReg <- append(lassoSplineReg, mean((mean(pred3) - y_test)^2)) # squared bias
lassoSplineReg <- append(lassoSplineReg, mean((pred3 - mean(pred3))^2))   # variance

# print table
results1 <- data.frame(lassoReg, lassoSplineReg, row.names = c('MSE', 'Squared-Bias', 'Variance'))
kable(results1,
      caption = "Results of Lasso Regression Models on Test Dataset",
      digits = 3) %>%
        kable_styling(latex_options = "hold_position")
# Run PCA on the continuous variables
Xstd <- train_set %>% select(temp, atemp, hum, windspeed)
pc_out <- prcomp(Xstd)

# Extend PCA to test set for holdout analysis
Xstd_test <- predict(pc_out, test_set)
pca_test <- cbind(test_set, Xstd_test) %>% select(-temp, -atemp, -hum, -windspeed)

# Create table to display variation
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) %>% kable
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")
```

```r
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")
# Create grid of NS DF to test
PC1_df <- c(rep(2, 625), rep(3, 625), rep(4, 625), rep(5, 625), rep(6, 625))
PC2_df <- c(rep(2, 125), rep(3, 125), rep(4, 125), rep(5, 125), rep(6, 125))
PC3_df <- c(rep(2,  25), rep(3,  25), rep(4,  25), rep(5,  25), rep(6,  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)

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

#Test each row from the grid in a loop
for(j in 1:nrow(Grid1)){
  # Create a temporary dataset to create folds and build model
  temp <- pca_train
  temp$holdoutpred <- rep(0, nrow(temp))
  for(i in 1:k){
    # Split temporary data into test (fold k) and train (folds not equal to k)
    train <- temp[folds$subsets[folds$which != i], ]
    val   <- temp[folds$subsets[folds$which == i], ]

    # Fit the linear model with the specified natural spline df for each cont. var
    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 +
                       poly(season, 2) +
                       holiday
                  , data = train)

    # Generate predictions on fold k
    newpred <- predict(pca_ns, newdata=val)

    # Save predictions in temp
    temp[folds$subsets[folds$which == i], ]$holdoutpred <- newpred
  }

  # Calculate RMSE from all folds combined
  RMSE <- round(sqrt(mean((temp$holdoutpred - temp$cnt)^2)), 4)

  # Save RMSE in grid 1
  Grid1[j, 6] <- RMSE
}

# Find the best model (min RMSE) from PC1, PC2 & PC3
```

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

# Create grid of NS DF to test
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)

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

#Test each row from the grid in a loop
for(j in 1:nrow(Grid2)){
  # Create a temporary dataset to create folds and build model
  temp <- pca_train
  temp$holdoutpred <- rep(0, nrow(temp))
  for(i in 1:k){
    # Split temporary data into test (fold k) and train (folds not equal to k)
    train <- temp[folds$subsets[folds$which != i], ]
    val   <- temp[folds$subsets[folds$which == i], ]

    # Fit the linear model with the specified natural spline df for each cont. var
    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 +
                   poly(season, 2) +
                   holiday
                 , data = train)

    # Generate predictions on fold k
    newpred <- predict(pca_ns, newdata=val)

    # Save predictions in temp
    temp[folds$subsets[folds$which == i], ]$holdoutpred <- newpred
  }

  # Calculate RMSE from all folds combined
  RMSE <- round(sqrt(mean((temp$holdoutpred - temp$cnt)^2)), 4)

  # Save RMSE in grid 2
  Grid2[j, 5] <- RMSE
}

# Find the best model (min RMSE) from PC1 & PC2
df_vec2 <- Grid2 %>% filter(RMSE == min(Grid2$RMSE))
```

```r
# Create grid of NS DF to test
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)

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

#Test each row from the grid in a loop
for(j in 1:nrow(Grid3)){
  # Create a temporary dataset to create folds and build model
  temp <- pca_train
  temp$holdoutpred <- rep(0, nrow(temp))
  for(i in 1:k){
    # Split temporary data into test (fold k) and train (folds not equal to k)
    train <- temp[folds$subsets[folds$which != i], ]
    val   <- temp[folds$subsets[folds$which == i], ]

    # Fit the linear model with the specified natural spline df for each cont. var
    pca_ns <- lm(cnt ~ ns(PC1, df = Grid3[j, 1]) +
                   ns(weekday, df = Grid3[j, 2]) +
                   ns(mnth, df = Grid3[j, 3]) +
                   weathersit +
                   yr +
                   poly(season, 2) +
                   holiday
                 , data = train)

    # Generate predictions on fold k
    newpred <- predict(pca_ns, newdata=val)

    # Save predictions in temp
    temp[folds$subsets[folds$which == i], ]$holdoutpred <- newpred
  }

  # Calculate RMSE from all folds combined
  RMSE <- round(sqrt(mean((temp$holdoutpred - temp$cnt)^2)), 4)

  # Save RMSE in grid 2
  Grid3[j, 4] <- RMSE
}

# Find the best model (min RMSE) from just PC1
df_vec3 <- Grid3 %>% filter(RMSE == min(Grid3$RMSE))

# Create Table of Best Models
df_vec2$PC3 <- 0
df_vec3$PC2 <- 0
df_vec3$PC3 <- 0
df_vec <- rbind(df_vec1, df_vec2, df_vec3)
```

```
Model <- c("PC1, PC2 & PC3", "PC1 & PC2", "PC1 only")
df_vec <- cbind(Model, df_vec)

# Print table
kable(df_vec, caption = "Best Model for Each Specification (Natural Spline DF)", digits = 1) %>%
  kable_styling(latex_options = "hold_position")

best <- df_vec %>% filter(RMSE == min(df_vec$RMSE))
PC1_df <- best[1, 2]
PC2_df <- best[1, 3]
PC3_df <- best[1, 4]
DOW_df <- best[1, 5]
Mth_df <- best[1, 6]

# Final PCA Model
pca_final <- lm(cnt ~ ns(PC1, df = PC1_df) +
                      ns(PC2, df = PC2_df) +
                      ns(PC3, df = PC3_df) +
                      ns(weekday, df = DOW_df) +
                      ns(mnth, df = Mth_df) +
                      weathersit +
                      yr +
                      poly(season, 2) +
                      holiday
                  , data = pca_train)

# Print final model regression output
pca_final %>% tidy() %>% kable(digits = 3)
# Generate predictions and holdout RMSE for the PCA Model
pca_pred <- predict(pca_final, newdata = pca_test)
pca_RMSE <- round(sqrt(mean((pca_pred - pca_test$cnt)^2)), 2)

# print table comparing all models on holdoutset
results <- data.frame(sqrt(ridgeReg[1]),
                      sqrt(ridgeSplineReg[1]),
                      sqrt(lassoReg[1]),
                      sqrt(lassoSplineReg[1]),
                      pca_RMSE, row.names = c('RMSE'))
colnames(results) <- c('Ridge', 'Ridge+Spline', 'Lasso', 'Lasso+Spline', 'PCA')

kable(results,
      caption = "Results of Models on Test Dataset",
      digits = 3) %>%
        kable_styling(latex_options = "hold_position")
library(tidyverse)
library(caret)
library(rsample)
library(kableExtra)
library(splines)
library(cvTools)
library(glmnet)
library(fastDummies)
library(ggrepel)
```

```
library(Metrics)
library(gridExtra)
library(splines)

# 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)

#### Create dataset with spline modified vars ####
# create splines for humidity
kn <- quantile(day_scaled$hum, probs = seq(.10, .90, by = .1)) # Define region boundaries
basis <- bs(day_scaled$hum,
                degree = 3,
                knots = kn) # Cubic splines basis functions
df <- data.frame(basis)
df <- df %>% rename_with( ~ paste0("hum_", .x))
day_scaled_spline <- merge(day_scaled, df, by.x=0, by.y=0)
day_scaled_spline <- day_scaled_spline %>% select(-c('Row.names', 'hum'))

# create splines for windspeed
kn <- quantile(day_scaled$windspeed, probs = seq(.10, .90, by = .1)) # Define region boundaries
basis <- bs(day_scaled$windspeed,
                degree = 3,
                knots = kn) # Cubic splines basis functions
df <- data.frame(basis)
df <- df %>% rename_with( ~ paste0("windspeed_", .x))
day_scaled_spline <- merge(day_scaled_spline, df, by.x=0, by.y=0)
day_scaled_spline <- day_scaled_spline %>% select(-c('Row.names', 'windspeed'))

# test train split
index <- initial_split(day_scaled_spline, prop = 0.75)
train_set_spline <- training(index)
test_set_spline <- testing(index)

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

```r
#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)
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)

humCnt <- ggplot(data=day) +
            geom_point(mapping=aes(x=hum, y=cnt)) +
            ggtitle("Impact of Humidity on Bike Rentals") +
            xlab("Normalized Humidity") + ylab("Count of Total Bike Rentals")
windCnt <- ggplot(data=day) +
            geom_point(mapping=aes(x=windspeed, y=cnt)) +
            ggtitle("Impact of Windspeed on Bike Rentals") +
            xlab("Normalized Windspeed") + ylab("Count of Total Bike Rentals")
grid.arrange(humCnt, windCnt, nrow=1)

# create datastructures for tuning process
y_train <- train_set$cnt
X_train <- as.matrix(train_set %>% select(-c('cnt', 'dteday')))
#X_train <- as.matrix(train_set[,-c(1:9)])

# create grid of potential hyperparam values, train model for each value using 5-fold cv
set.seed(54321)
grid <- 2^seq(-6, 12, length = 100)
cv_out1 <- cv.glmnet(x = X_train, y = y_train,
                     type.measure='mse',
                     nfolds = 5,
                     alpha = 1,
                     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_out1$glmnet.fit$beta)
lambdas1 = cv_out1$lambda
names(lambdas1) = colnames(betas)
coefPlot <- as.data.frame(betas) %>%
              tibble::rownames_to_column("variable") %>%
              pivot_longer(-variable) %>%
              mutate(lambda=lambdas1[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
```

```r
# display the output of 5-fold cv for ridge regression comparing MSE to lambdas
plot(cv_out1)

# create datastructures for tuning process
y_train_spline <- train_set_spline$cnt
X_train_spline <- as.matrix(train_set_spline %>% select(-c('cnt', 'dteday')))

# create grid of potential hyperparam values, train model for each value using 5-fold cv
set.seed(54321)
grid <- 10^seq(-4, 8, length = 100)
cv_out3 <- cv.glmnet(x = X_train_spline, y = y_train_spline,
                     type.measure='mse',
                     nfolds = 5,
                     alpha = 1,
                     lambda = grid)

plot(cv_out3)

# evaluate performance of ridge regression model
y_test <- test_set$cnt
X_test <- as.matrix(test_set %>% select(-c('cnt', 'dteday')))
pred1 <- predict(cv_out1, X_test)
lassoReg <- c()
lassoReg <- append(lassoReg, mean((pred1 - y_test)^2))       # mse
lassoReg <- append(lassoReg, mean((mean(pred1) - y_test)^2)) # squared bias
lassoReg <- append(lassoReg, mean((pred1 - mean(pred1))^2))   # variance

# evaluate performance of ridge+spline regression model
y_test <- test_set_spline$cnt
X_test <- as.matrix(test_set_spline %>% select(-c('cnt', 'dteday')))
pred3 <- predict(cv_out3, X_test)
lassoSplineReg <- c()
lassoSplineReg <- append(lassoSplineReg, mean((pred3 - y_test)^2))       # mse
lassoSplineReg <- append(lassoSplineReg, mean((mean(pred3) - y_test)^2)) # squared bias
lassoSplineReg <- append(lassoSplineReg, mean((pred3 - mean(pred3))^2))   # variance

# print table
results1 <- data.frame(lassoReg, lassoSplineReg, row.names = c('MSE', 'Squared-Bias', 'Variance'))
kable(results1,
      caption = "Results of Lasso Regression Models on Test Dataset",
      digits = 3) %>%
        kable_styling(latex_options = "hold_position")

# 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
set.seed(54321)
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

plot(cv_out)

# create datastructures for tuning process
y_train_spline <- train_set_spline$cnt
X_train_spline <- as.matrix(train_set_spline %>% select(-c('cnt', 'dteday')))

# create grid of potential hyperparam values, train model for each value using 5-fold cv
set.seed(54321)
grid <- 10^seq(-2, 10, length = 100)
cv_out2 <- cv.glmnet(x = X_train_spline, y = y_train_spline,
                    type.measure='mse',
                    nfolds = 5,
                    alpha = 0,
                    lambda = grid)

plot(cv_out2)

# evaluate performance of ridge regression model
y_test <- test_set$cnt
X_test <- as.matrix(test_set %>% select(-c('cnt', 'dteday')))
pred <- predict(cv_out, X_test)
ridgeReg <- c()
ridgeReg <- append(ridgeReg, mean((pred - y_test)^2))       # mse
ridgeReg <- append(ridgeReg, mean((mean(pred) - y_test)^2)) # squared bias
ridgeReg <- append(ridgeReg, mean((pred - mean(pred))^2))   # variance

# evaluate performance of ridge+spline regression model
y_test <- test_set_spline$cnt
X_test <- as.matrix(test_set_spline %>% select(-c('cnt', 'dteday')))
pred <- predict(cv_out2, X_test)
ridgeSplineReg <- c()
ridgeSplineReg <- append(ridgeSplineReg, mean((pred - y_test)^2))       # mse
ridgeSplineReg <- append(ridgeSplineReg, mean((mean(pred) - y_test)^2)) # squared bias
ridgeSplineReg <- append(ridgeSplineReg, mean((pred - mean(pred))^2))   # variance
```

```r
# print table
results <- data.frame(ridgeReg, ridgeSplineReg, row.names = c('MSE', 'Squared-Bias', 'Variance'))
kable(results,
      caption = "Results of Ridge Regression Models on Test Dataset",
      digits = 3) %>%
        kable_styling(latex_options = "hold_position")

# Run PCA on the continuous variables
Xstd <- train_set %>% select(temp, atemp, hum, windspeed)
pc_out <- prcomp(Xstd)

# Extend PCA to test set for holdout analysis
Xstd_test <- predict(pc_out, test_set)
pca_test <- cbind(test_set, Xstd_test) %>% select(-temp, -atemp, -hum, -windspeed)

# Create table to display variation
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))
```