STAT 685: Dr. Suojin Wang's Group

Modeling Seoul Bike Sharing Demand

Nam Tran, Bai Zou

October 22, 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Our data set is the “Seoul Bike Sharing Demand Data Set”, which on a high level contains hourly data for bike usage as well as various covariates that might be useful, e.g., temperature. Further, it contains around one year of data.

The data set has been aggregated and been uploaded to the UCI Machine Learning Repository, located here: http://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand

At first glance, relevant pieces are:

- Contains 8760 observations
- There are 14 columns

Regarding motivation for the data set and its potential use, the following is taken from the UCI website and was attached by the team that donated the data: ”

”Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. It is important to make the rental bike available and accessible to the public at the right time as it lessens the waiting time. Eventually, providing the city with a stable supply of rental bikes becomes a major concern. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes.

The dataset contains weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour and date information.”

1.2 Previous Work

There are two papers dealing with this data set explicitly. We're including both papers and in addition providing high level commentary on the paper and our concerns.

1.2.1 Paper 1: VE, Park, and Cho (2020, Computer Communications), Using Data Mining Techniques

High Level:

- Attempted to predict the hourly demand.
- Claims .96 R^2 and .92 R^2 for train and test, respectively.
- Uses "Boruta" to identify important features, which underneath the hood uses random forest as part of the algorithm.
- Considered linear regression, GBM, SVM's (radial basis function), boosted trees, and xgboost; had a nice mathematical summary of each.

Thoughts/Concerns

- Didn't consider L1 Regularization, i.e., LASSO, which seems the most obvious for feature importance/selection/ranking. For example, do the full path as we go over λ and see the order they get dropped, where the sooner they drop the less important they are.
- Make no reference to how they split up into 75% train and 25% test, e.g., is it interleave of a specific calendar day and everything after is post? If interleaved, the the 75% train's distribution and 25% test's distribution are effectively identical and learning on the train portion is considered "cheating" since data has leaked.
- Didn't consider non-linear transforms of the data, which may not be that important given the usage of decision trees, but could have allowed plain linear regression to perform better.

1.2.2 Paper 2: VE and Cho (2020, European Journal of Remote Sensing), A Rule-Based Model

High Level

- Consider CUBIST, regularized random forest, CART, KNN, Conditional Inference Tree.
- Almost identical to previous paper.

- They had an additional data source, the “Capital Bikeshare program,” for which the dataset came from Kaggle. It didn’t seem that they used this in an “interesting” way (interleaving the data, using it as a validation) but instead just considered it as another data source for which they ran their same methodology and if they got the same R^2 and other metrics, then they’d consider that would be successful.

Thoughts/Concerns

- Still don’t talk about train/test split methodology.
- The last bullet point of high level thoughts.

1.3 Scope and Goal

Based on the description above, we break it into two potential business requirements here:

- Predict next day hourly demand based on historical data until the current day.
- Real-time prediction for next hour demand based on historical data until the current hour.

The scope in this study is to:

- Re-define training and testing data with anchor time.
- Re-evaluate estimation methods with data splitting by anchor time.
- Build forecasting models to predict the next hour demand and compare the models in terms of prediction accuracy, prediction variance, running time, etc.

Chapter 2

Data Exploratory

2.1 Seoul Bike Sharing Demand Data

- Downloaded the data from the UCI Machine Learning Repo.
- Contains 8760 measurements of number of bikes rented over 364.958 days.

Features of the data are,

- `DateTime`
- `RentedBikeCount`
- `Temp`, in Celsius.
- `Humidity`, in percent, max of 100.
- `Windspeed`
- `Visibility` out to 10 meters.
- `DewPointTemp`, in Celsius.
- `SolarRadiation`
- `Rainfall`, in mm.
- `Snowfall`, in cm.
- `Seasons`, a factor with levels {Winter, Spring, Summer, Autumn}.
- `Holiday`, a factor with levels {Holiday, No holiday}.
- `FunctionalDay`, a factor with levels {NoFunc(Non Functional Hours), Fun(Functional hours)}

2.2 Time Series Data

Fundamentally, our data is time-series data (Figure 2.1). As such, let y_t be the time series we're working to model, i.e., Seoul's bike sharing data.

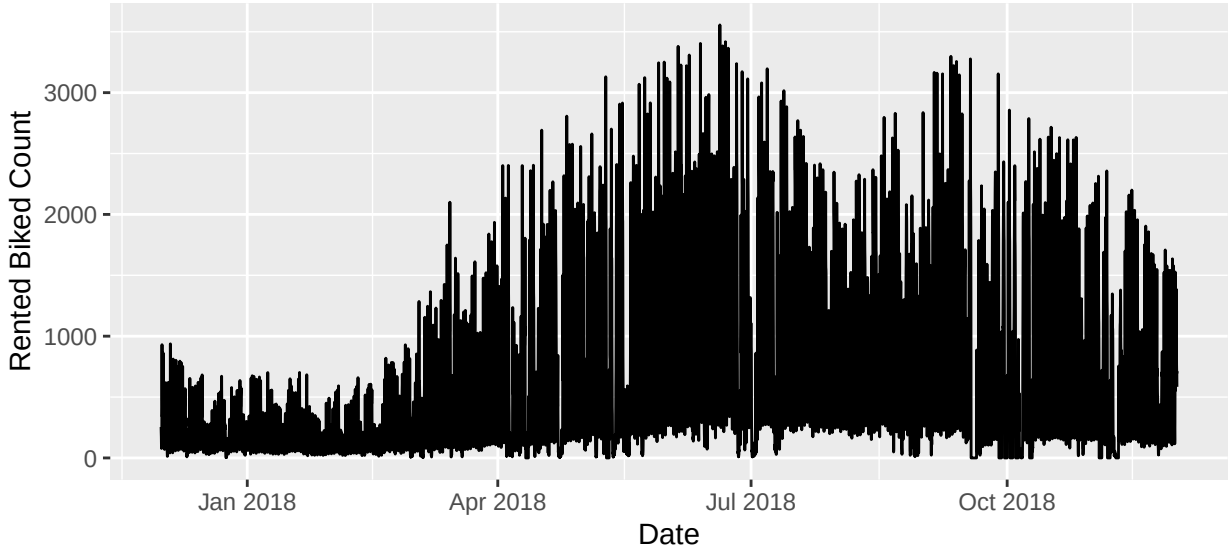

Figure 2.1: Hourly Rented Bike Count Over Entire Time Period

2.2.1 Stationarity

It's arguable that there might be a strong seasonality component (less in winter more in summer), but that's hard to ascertain here since we only have one year's of data and only have one cycle. Further, there might be strong seasonality on an intraday basis (less in early morning and ramp up afterwards). If there was a strong seasonality component, we'd say our data isn't stationary, since on a first order basis, $E(y_t)$ will be dependent on t. Stationarity is important for a multitude of reasons, including *averaging being meaningful* and any *conditional expectation model we build is stable*.

Note, we can still incorporate terms to make a time series stationary, e.g., trend-stationary.

We can test this directly using the Augmented Dickey-Fuller (ADF) Test, which intuitively tests for the presence of a unit root, which implies non-stationarity. H_0 for ADF is that y_t is non-stationary, and H_a is that y_t is stationary. Note there are different types of stationarity, e.g., in presence of drift (μ) or linear trend (βt).

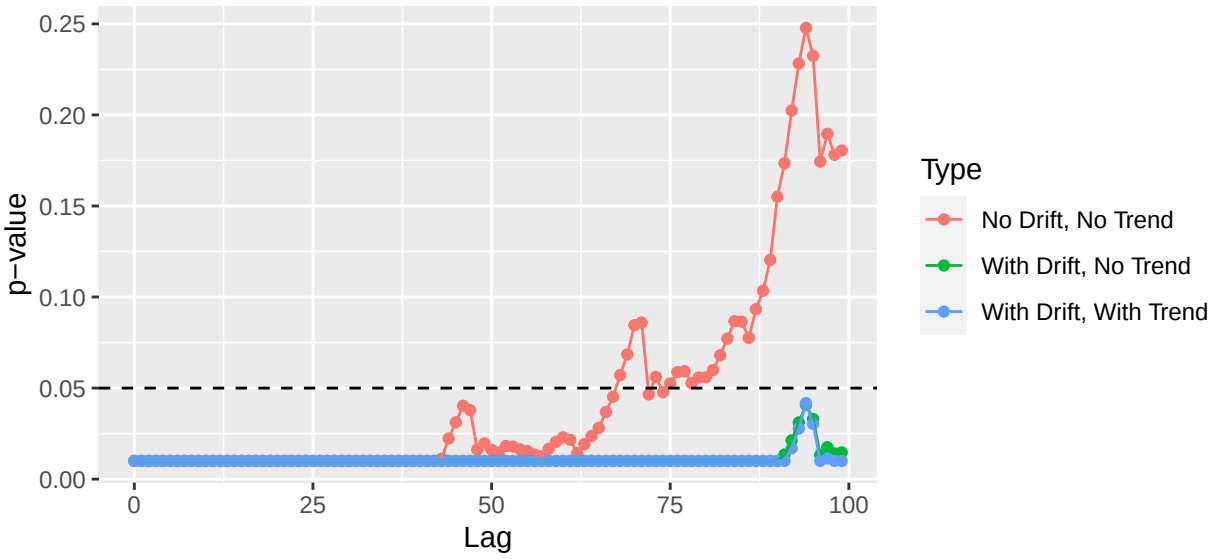

Figure 2.2: Augmented Dickey Fuller (ADF) Test for Stationarity

Note that each lag, i.e., tick mark, in the ADF figure represents an hour. Under the most relaxed condition of no drift and no trend, then we can see that we start getting significant non-stationarity post lag 48, which represents approximately two days past. While this can be handled by differencing, as suggested by the stationarity for more restrictive conditions, this can also suggest that we can include lagged covariates of the response, i.e., lagged y_t, which we will ascertain next when looking at the auto-correlation function (ACF) plots and the partial auto-correlation function (PACF) plots.

2.2.2 Autocorrelation and Partial Autocorrelation of Rented Biked Count, y_t

The ACF looks at correlation of y_t with lagged versions of itself, e.g., y_{t-k}. The PACF differs in that it looks at correlation of y_t with lagged versions of itself, e.g., y_{t-k}, while controlling for the intermediary lags, e.g., $\tilde{y} = \{y_{t-1}, y_{t-2}, \dots, y_{t-k+1}\}$. From a practical standpoint, when considering PACF, we regress y_t on $\tilde{y}$ and y_{t-k} on $\tilde{y}$, and then look at the correlation of their respective residuals.

Here, We look at the ACF and PACF of y_t up to 100 and 50 lags.

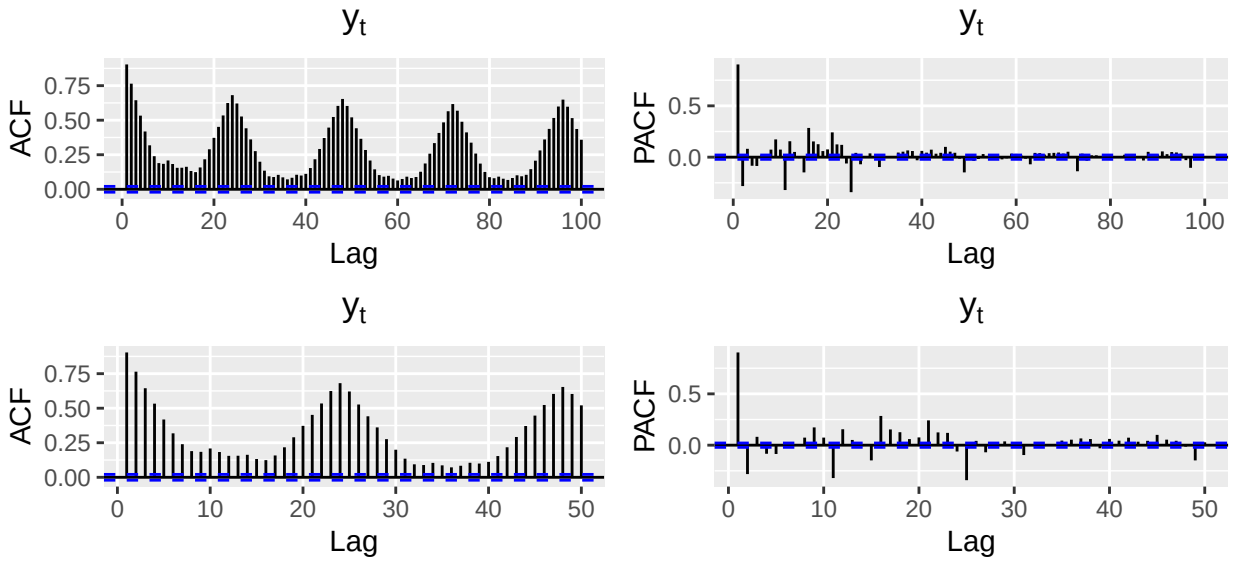

Figure 2.3: Autocorrelation and Partial Autocorrelation of Rented Bike Count, i.e., y_t

Math theory states that an AR(p) model would have a hard cutoff to zero in the PACF plot for $h > p$, and a MA(q) model would have a hard cutoff to zero in the ACF plot for $h > q$. From the ACF plot and seeing statistically significant autocorrelations all the way out, a simple MA(q) model will not suffice. Looking at the PACF plot, we see a strong "cut-off" at around lag 25, suggesting an AR(25) model. Needless to say, an AR(25) model isn't very palatable and doesn't seem parsimonious. As such, we seemingly can't get away with a simple MA(q) nor a simple AR(p) model.

While we can't get a simple AR(p)$orMA$(q$) model, we can still use the results of the ACF and PACF plots to suggest that we need lagged values of our supervisor as additional covariates.

2.3 Feature Attributes

2.3.1 Hourly Trend

Plot below is showing the mean hourly demand by season. It is clear that winter season has much lower demand and summer season has relatively higher demand. Hourly trend is similar in each season with two peak time per day - 8 AM and 6 PM. The hour information could be used as either qualitative or quantitative since demand is not linearly related to hour.

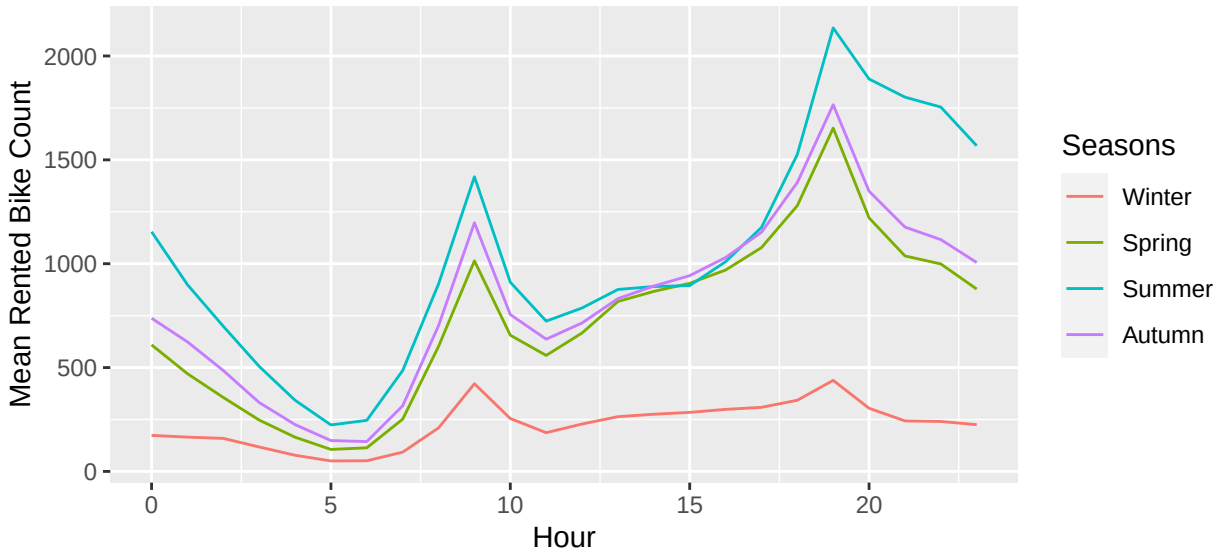

Figure 2.4: Rented Bike Count by Hour Grouped by Seasons

2.3.2 Qualitative Variables

- The plots shows more rented bike count in non-holidays than holidays except for summer (Figure 2.5).
- If functional day is “no”, there’s no any bike rented (Figure 2.6).
- Day of week is not making significant difference in rented bike count (Figure 2.7).

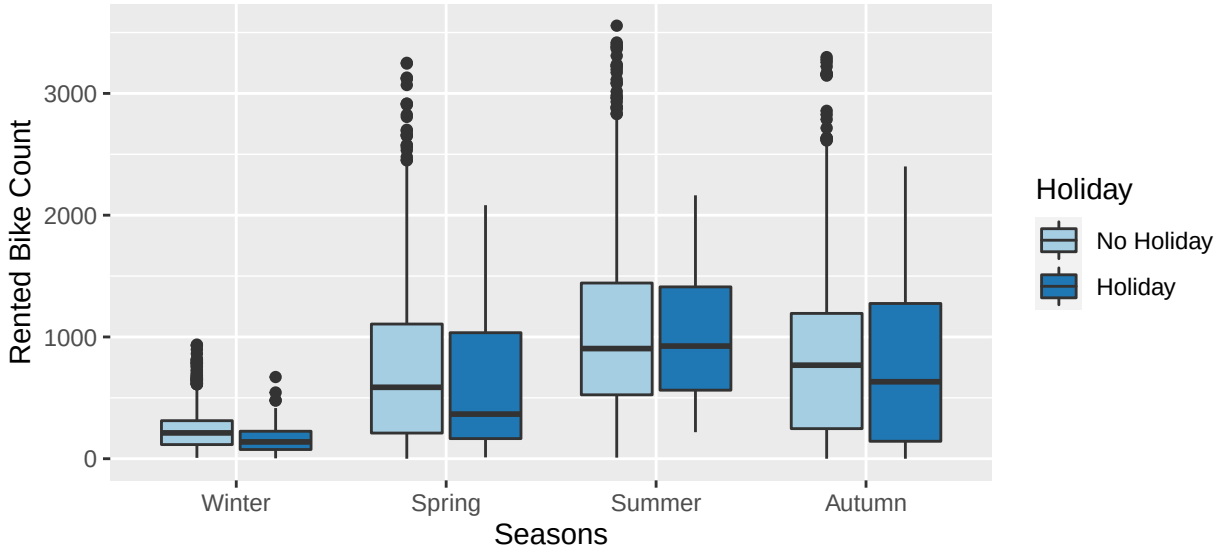

Figure 2.5: Rented Bike Count by Season Grouped by Holiday

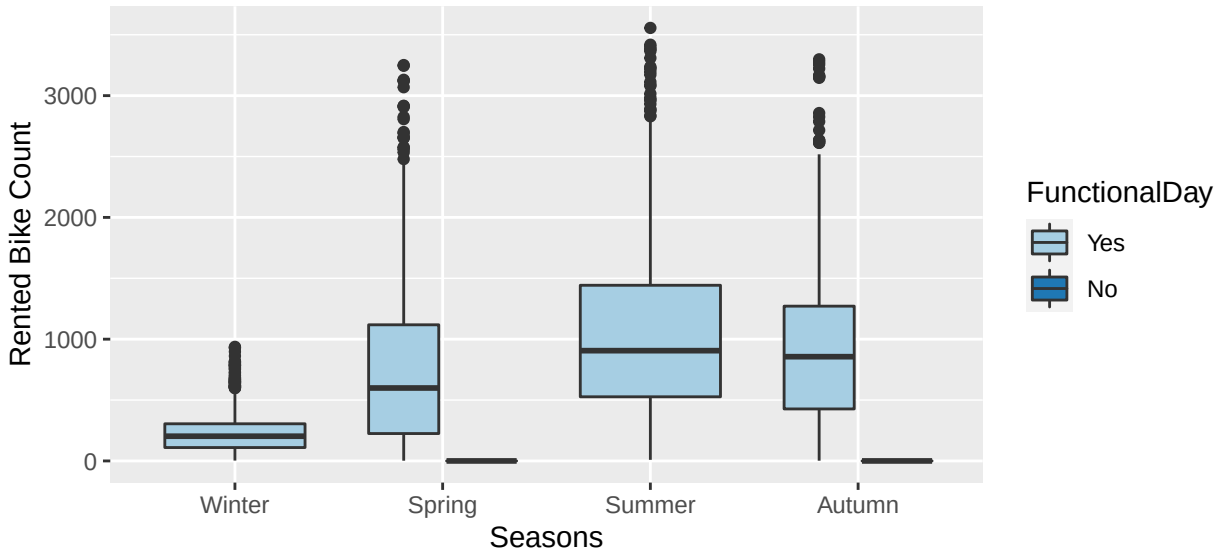

Figure 2.6: Rented Bike Count by Season Grouped by Functional Day

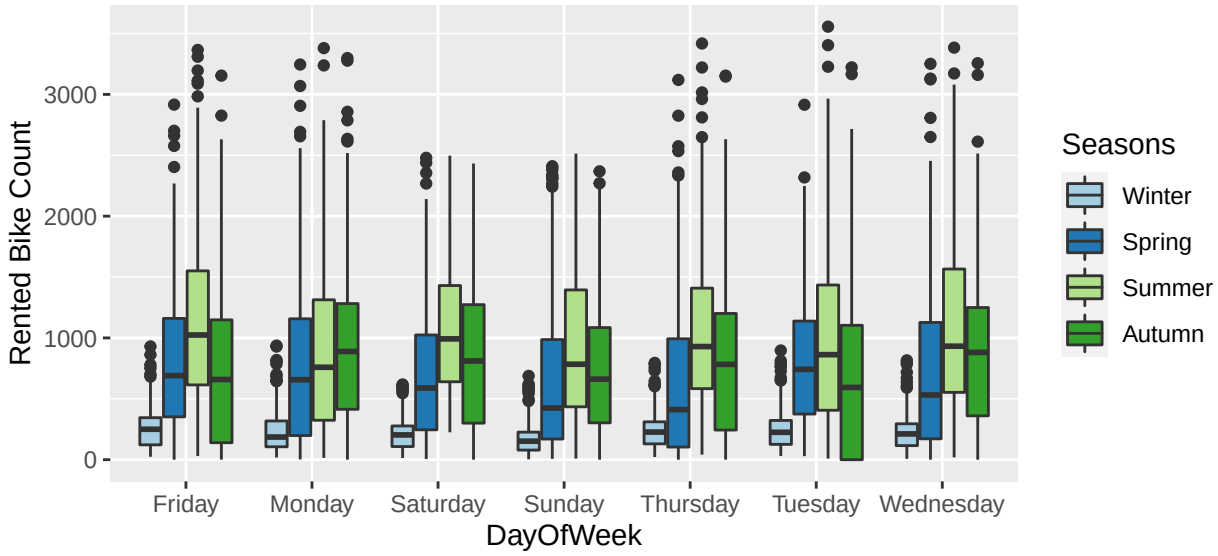

Figure 2.7: Rented Bike Count by Day of Week Grouped by Season

2.3.3 Quantitative Variables

Figure 2.8 and Figure 2.9 are showing correlations between quantitative variables and demand:

- The covariance matrix shows Temp, Hour has relatively higher correlation with RentedBike-Count (>0.4).
- DewPointTemp and SolarRadiation have correlation greater than 0.2.
- Temp and DewPointTemp are highly correlated (0.9).

- **No clear linear relationship can be identified between response variable and quantitative Variables**

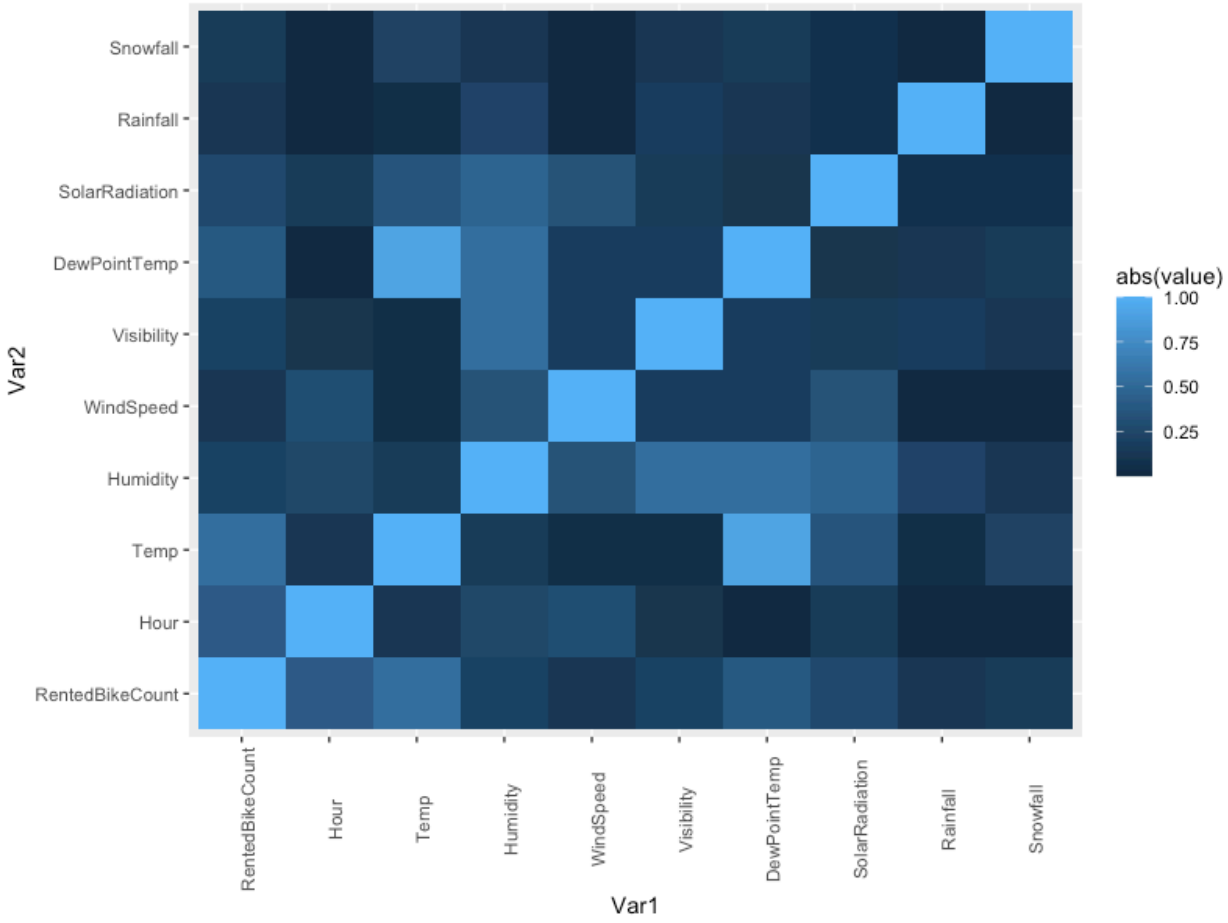

Figure 2.8 Variable Correlation Matrix

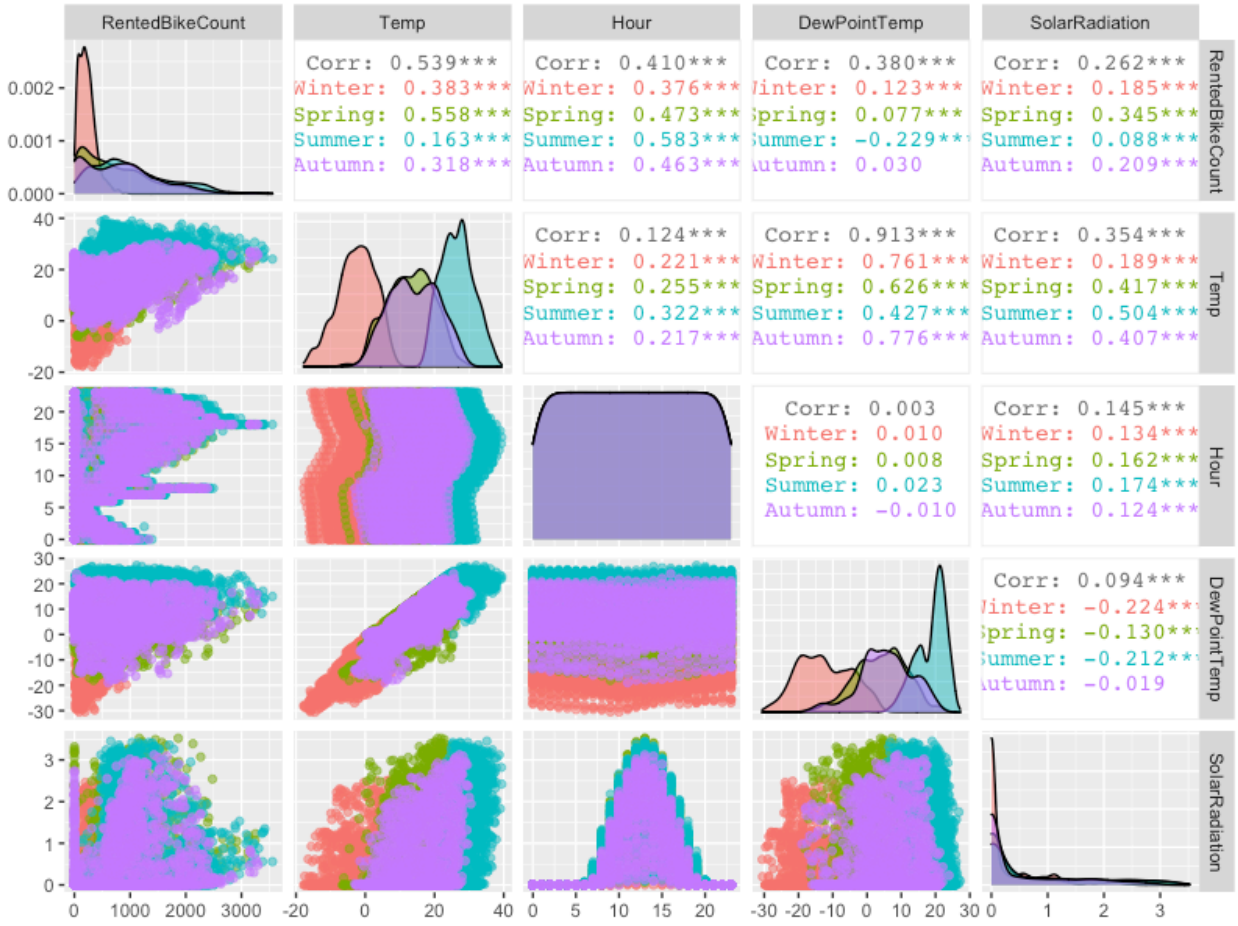

Figure 2.9 Top 4 Variable Correlation Matrix

2.4 Splitting Training and Testing Data

The data set includes one year hourly bike rented count from Dec 2017 to Nov 2019. Splitting training and testing data in any anchor date will cause incomplete yearly distribution and information loss in training data. For example, there are only two days' observations for non-functional day before September 2018, which leaves little evidence for the model to identify the impact of Functional Day during training process if setting anchor date prior to September.

To minimize the information loss and maximize the training data size available, the testing anchor date and time will be set no earlier than November 1, 2018:

- Feature distributions are close to all year distribution (See 2.4.1 and 2.4.2).
- Preliminary model testing shows the best result when using Nov 2018 data for testing (Chapter 3).

2.4.1 Weather Information Distribution

Figure 2.10 and 2.11 below are comparing distributions of some weather features for all observations and subset of observations before September 1, October 1 and November 1, 2018. In general, the last subset (setting anchor day at November 1, 2018) has a close enough distribution comparing the one year data set.

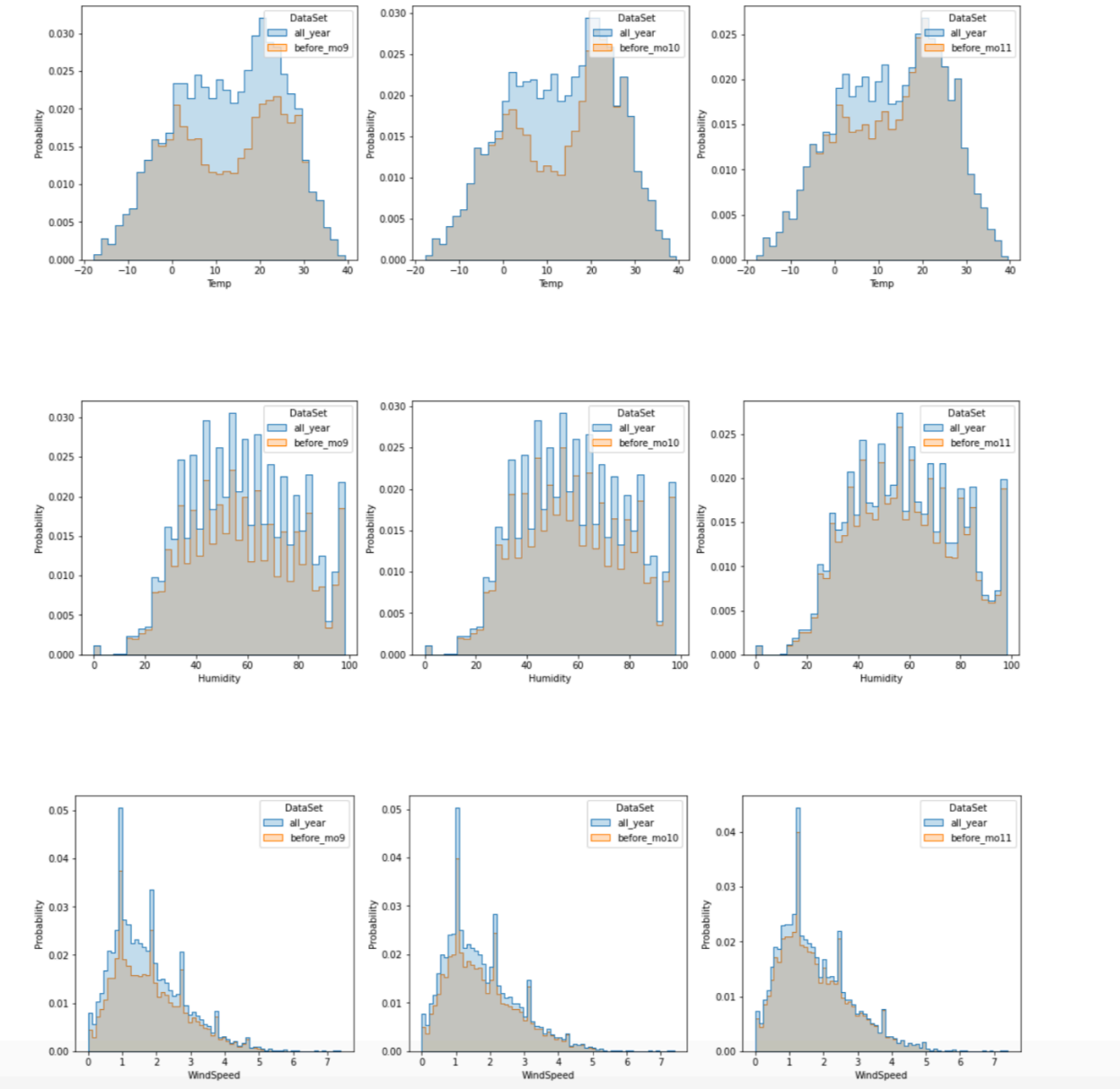

Figure 2.10 Temp, Humidity and WindSpeed Distribution

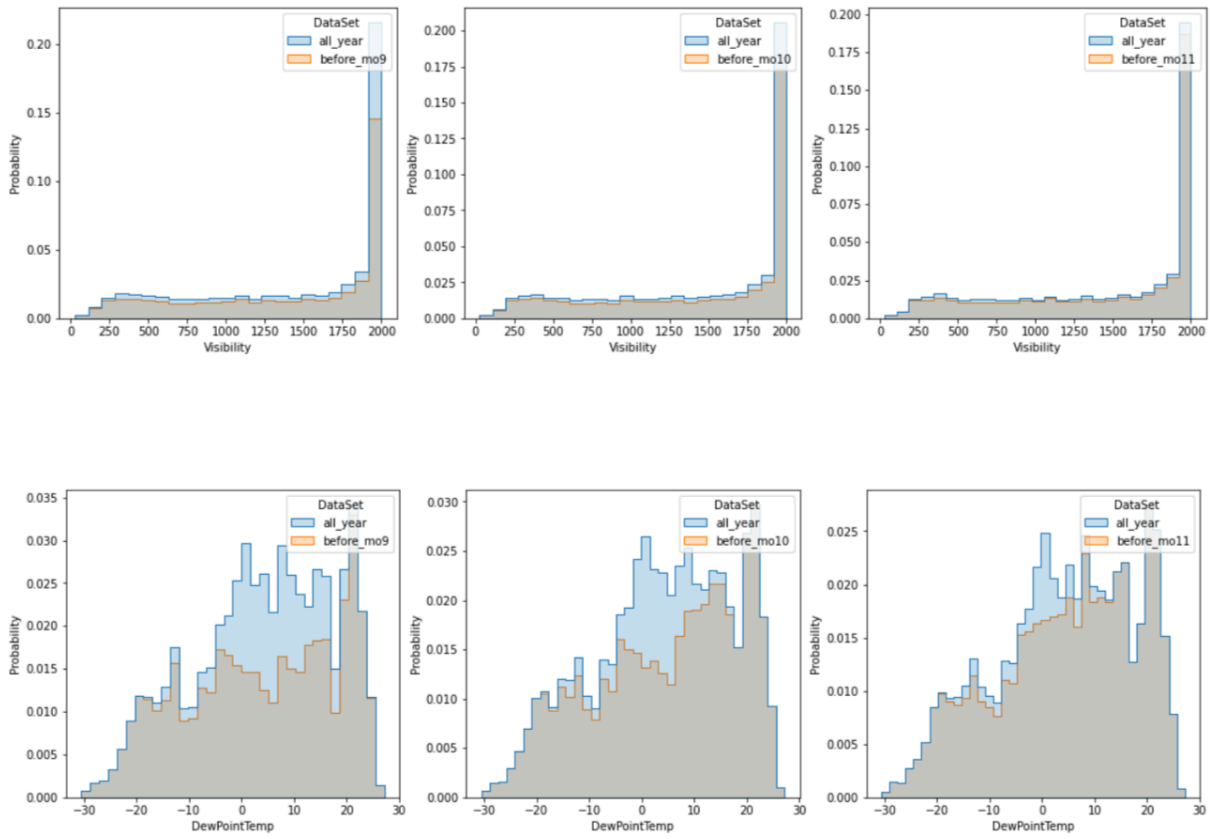

Figure 2.11 Visibility and DewPointTemp Distribution

2.4.2 Function Day and Holiday Distribution

Tables below are showing number of observations by category in each data set. The last data set (setting anchor day at November 1, 2018) has the closest percentage comparing to the one year data.

Table 2.1 Number of Observations by Season

	all_year	before_mo9	before_mo10	before_mo11
Autumn	0.25	NaN	0.1	0.18
Spring	0.25	0.34	0.3	0.27
Summer	0.25	0.34	0.3	0.27
Winter	0.25	0.33	0.3	0.27

Table 2.2 Number of Observations by Holidays

	all_year	before_mo9	before_mo10	before_mo11
No Holiday	0.95	0.95	0.95	0.95
Holiday	0.05	0.05	0.05	0.05

Table 2.3 Number of Observations by Function Day

	all_year	before_mo9	before_mo10	before_mo11
Yes	0.97	0.99	0.98	0.97
No	0.03	0.01	0.02	0.03

Chapter 3

Estimation Methods Comparison

3.1 Theoretical Methodology

3.1.1 Data

We have a collection of observations housed in a matrix X that is nxp, i.e., n observations and p covariates. Further, for each observation, we have an associated supervisor value, for which all the supervisor values are housed in a column vector y that is nx1, i.e., n supervisor values for each of the n associated observation.

3.1.2 Risk

We want to use training data and different algorithms to produce a $\hat{f} : \mathbb{R}^p \mapsto \mathbb{R}$, such that we can make predictions with $\hat{f}$, i.e., $\hat{f}(X) = \hat{y}$, where $x \in \mathbb{R}^p$ and $\hat{y} \in \mathbb{R}$, such that $\hat{y}$ is a "good" prediction of y, the unobserved supervisor.

One way to define "good" is to define it in the context of "loss", specifically $\ell(\hat{y}, y)$. There are a multitude of loss functions to consider, but a "popular" loss for regression is the squared error loss, i.e., $\ell(\hat{y}, y) = (\hat{y} - y)^2$, where deviations from the true y value is penalized in a squared fashion.

We define "good" to be the risk for f, namely $R(f) = \mathbb{E}\ell(f(X), Y)$, noting that X, Y are random variables **but** $R(f)$ isn't random, due to the expectation. In practice, we can use "test error" as an estimate for the risk. We can also use "cross-validation" as another estimate for the risk as well.

3.1.3 Identifying f_* vs. $\widehat{f}$

Let $f_* = \arg\min_f R(f)$, i.e., f_* has the lowest risk amongst the entire family of possible f. But, f_* is theoretical, since we don't know the entire joint distribution of (X, Y). As such, $\widehat{f}$ is our best guess of f_*.

3.1.4 Summary

In short, our goal is to consider different algorithms and make the best predictions we can, as defined by jointly by our risk estimate and the embedded loss metric, squared error loss $\ell(\widehat{y}, y) = (\widehat{y} - y)^2$ in our case. We will estimate risk in two different ways, cross-validation to help guide our hyper-parameter selection, and "test error" as the final hold-out to evaluate the "tuned" hyper-parameters.

3.2 Linear Methods

3.2.1 Linear Regression

Let $\beta^T, x^T \in \mathbb{R}^p$, then we wish to model y as $y = x^T\beta + \epsilon$, i.e., we want to project y onto the subspace spanned by X. In short, $\widehat{\beta} = \arg\min_\beta ||y - X\beta||_2^2$.

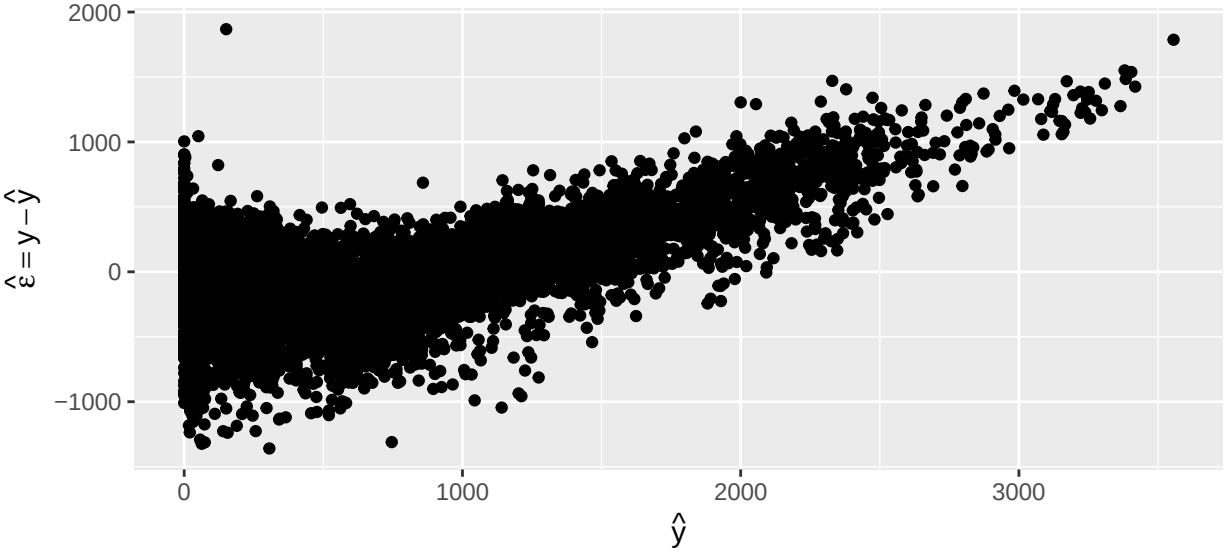

Figure 3.1: Residual Plot for Linear Regression

From this residual plot, it suggests that multiple linear regression isn't appropriate for this data set. Namely, in an idealized setting, we wouldn't notice any distinct patterns in the residuals, but in this case, we see a clear increase in residual values as our estimate $\widehat{y}$ gets larger. This may be due to the

supervisor being count data, for which a Poisson regression or applying a square-root transform to the supervisor.

To be more specific, for us to do inference using Linear Regression, independent of prediction, we need to have $\mathbb{E}\epsilon = 0$, $\mathbb{V}(\epsilon)$, and $\text{Cov}(\epsilon_i, \epsilon_j) = 0$, for which all three conditions aren't satisfied. This isn't a problem specifically for us, since we care about prediction, but it does suggest model misspecification.

3.2.2 Elastic Net

Elastic Net is an extension of Linear Regression, where we do a mixture of both of both L_1 regularization (penalty of $||\beta||_1$) and L_2 regularization (penalty of $||\beta||_2^2$). Then, $\widehat{\beta}(\lambda_1, \lambda_2) = \arg\min_\beta ||y - X\beta||_2^2 + \lambda_1||\beta||_1 + \lambda_2||\beta||_2^2$.

Note most parameterizations of Elastic Net instead of having separate λ_1 and λ_2 have a singular λ and a "mixing ratio" between L_1 and L_2 regularization in the form of α. Then, $\widehat{\beta}(\lambda, \alpha) = \arg\min_\beta (||y - X\beta||_2^2 + \lambda((1-\alpha)||\beta||_2^2 + \alpha||\beta||_1))$. In this parameterization, note that $\alpha = 1$ results in LASSO, which has only L_1 regularization , and that $\alpha = 0$ results in Ridge Regression, which has only L_2 regularization.

An open question remains though on how to choose α, the mixture between L_1 and L_2 regularization, and λ, how much penalty to impose. For this, we can use k-fold cross-validation risk estimates. Namely, for a particular set of hyper-parameters we wish to consider, we can take our training data and split it into k chunks, and for each chunk, we hold it out as the "test" data and learn on the remaining $k-1$ chunks (using our chosen hyper-parameters) and then evaluate on the k^{th} holdout using, for example, squared error, i.e., $\sum_{i=1}^n (y_i) - \widehat{y}_i)^2$. We would then average over all k risk estimatess and come up with a singular risk estimate, i.e., the cross-validation risk estimate for the hyper-parameters we used. Then, for the entire set of hyper-parameters, we'd have an associated cross-validation risk-estimate and consequently would choose the one with the lowest risk-estimate, which we're trying to minimmize.

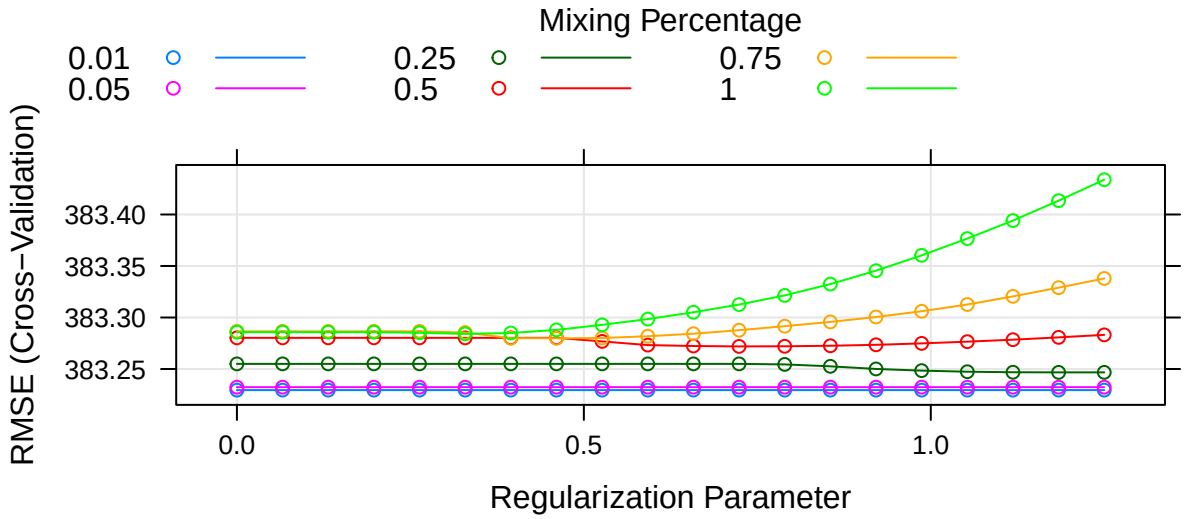

Figure 3.2: Cross-Validation Risk Estimates for Elastic Net Hyper Parameters

Looking at the cross-validation risk estimates, the minimal test error is at $\hat{\alpha} = 0.01$ and $\hat{\lambda} = 1.25$, suggesting that we prefer minimal L_1 regularization ($\alpha = 0$ is strictly L_2 reguarlization).

Note we've run into a boundary condition, i.e., we don't know if having $\hat{lambda} > 1.25$ will result in an even lower risk estimate. As such, we expand past the boundary and see if we can get a lower risk estimate.

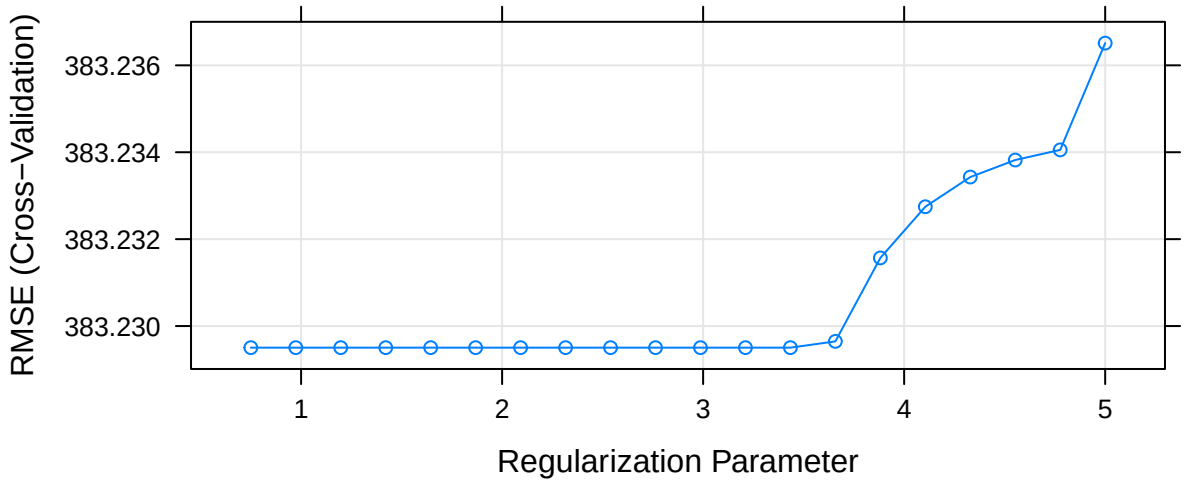

Figure 3.3: Cross-Validation Risk Estimates for Elastic Net Hyper Parameters Past Boundary Condition

Thus, the hyper-parameters that minimize the cross-validation risk estimate is $\hat{\alpha} = 0.01$ and $\hat{\lambda} = 3.4342105$.

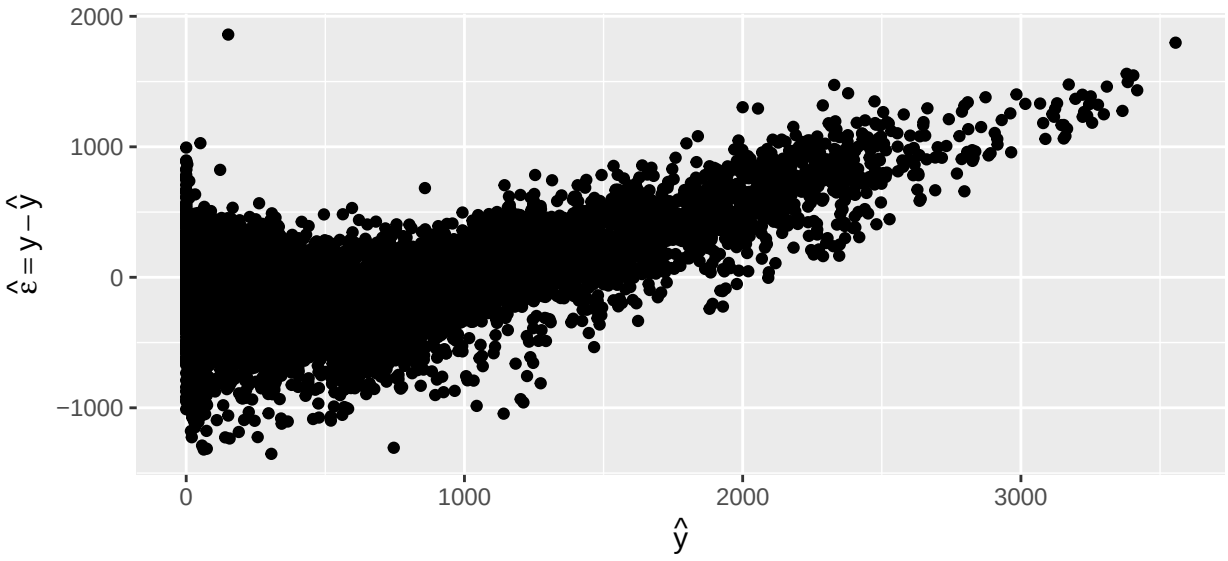

Figure 3.4: Residual Plot for Elastic Net

Not surprisingly, the residual plot for Elastic Net shows similar behavior to Linear Regression, suggesting that Elastic Net isn't an appropriate model and further confirmation the linear models aren't appropriate for the problem we have.

3.2.3 Refitted Lasso

3.2.3.1 Theory

Note lasso does both model selection as well as implicitly does regularization, which can produce a solution that has too much bias. To overcome this but still attain the benefits of model selection, we can choose a λ_{1SE}, i.e., the "one-standard-error" λ and then refit with unreguarlized linear regression.

3.2.3.2 Example

```
set.seed(1)

# Fitting with glmnet in general with alpha = 1.0
glmnetOut     = cv.glmnet(x = xlFullMat, y = yl, alpha = 1.0)

# We use lambda.min since lambda.1se will result in an empty active set!
betaHatGlmnet = coef(glmnetOut, s = "lambda.min")
refittedSIdx = which(as.vector(abs(betaHatGlmnet) > 1e-6)[-1])
```

```
# Fitting an unregularized linear model
refittedLassoOut = train(x = xlFullMat[,refittedSIdx], y = yl,
                         method = "lm",
                         trControl = trControl)
```

```
yhatRefitted = predict(refittedLassoOut, xlFullMat[,refittedSIdx])
```

3.3 Non-Linear Methods

3.3.1 Multivariate Adaptive Regression Splines (MARS)

3.3.2 Decision Tree

3.3.3 Random Forest

3.3.4 Boosting

3.3.4.1 Theory

Fitting, in a stepwise fashion, a greedy nonparametric procedure. Meta-parameters are λ, the learning rate, and B the number of steps.

3.3.4.2 Example

Chapter 4

Forecasting Model Comparison

There are three models considered to predict the next hour demand.(Assume the current day is X, and current hour is Y).

- Model 1: One-time prediction for day X hourly demand by end of day X-1 with data updated to day X-1.
- Model 2: Real-time model training to predict next hour (Y+1) demand with data updated to day X hour Y.
- Model 3: Continuous model training based on Model 1 and updated data from hour 1 to Y on day X.

Testing data setting:

- Randomly select n(5 - 10 depends on run time) days as anchor days from Nov 2018:
  - For each anchor day, all observations prior to this anchor day is considered as training data.
  - For each anchor day, the next 24 hour observations is considered as testing data for Model 1.
  - There are n sets of training and testing data sets.
- Each anchor day selected above will have 24 sets of training and testing data based on anchor hour (Model 2 & Model 3).
- Or randomly select m anchor hours for each anchor day selected above (if run time becomes a consideration).

4.1 Model 1: One-time Prediction

4.2 Model 2: Real-time Model Training

4.3 Model 3: Continuous Model Training

4.4 Model Comparison

Chapter 5

Result and Conclusion