STAT 685: Dr. Suojin Wang's Group

Modeling Seoul Bike Sharing Demand

Nam Tran, Bai Zou

November 12, 2020

Contents

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

E., Park, and Cho [2] looked at the same bike sharing data and looked at different learning algorithms on the original covariates in order to optimize prediction, using squared error as their evaluation metric. Different learning algorithms they look at include linear regression, gradient boosting machine's, support vector machines with radial basis functions, as well as xgboost. Ultimately, their best results were $R^2_{\text{train}} = 0.96$ and $R^2_{\text{test}} = 0.92$ using xgboost.

E. and Cho [1] only varies marginaly from E., Park, and Cho [2] in that they have an additional data set they consider, which they don't use as additional test data but instead run their same methodology on the Seoul bike sharing data and seeing if they get similar results. They do look at different learning algorithms, including CUBIST, random forest, CART, KNN, and conditional inference trees.

We have a deep concern with the previous work in that they don't make no reference to how they split up into 75% train and 25% test, e.g., is it randomly choose interleaved train and test or is it a specific calendar day and everything after is test and everythign prior is train? If interleaved, the the 75% training data's distribution and 25% test data's distribution are effectively identical and learning on the train portion can be deemed "cheating" since data has leaked.

Also, from a learning methodology, they don't consider L_1 regularization directly when using linear regression. Further, they don'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.

1.3 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.
- Improve the forecasting by considering dependency of previous demand.

- Test forecasting models in real business requirements.

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.

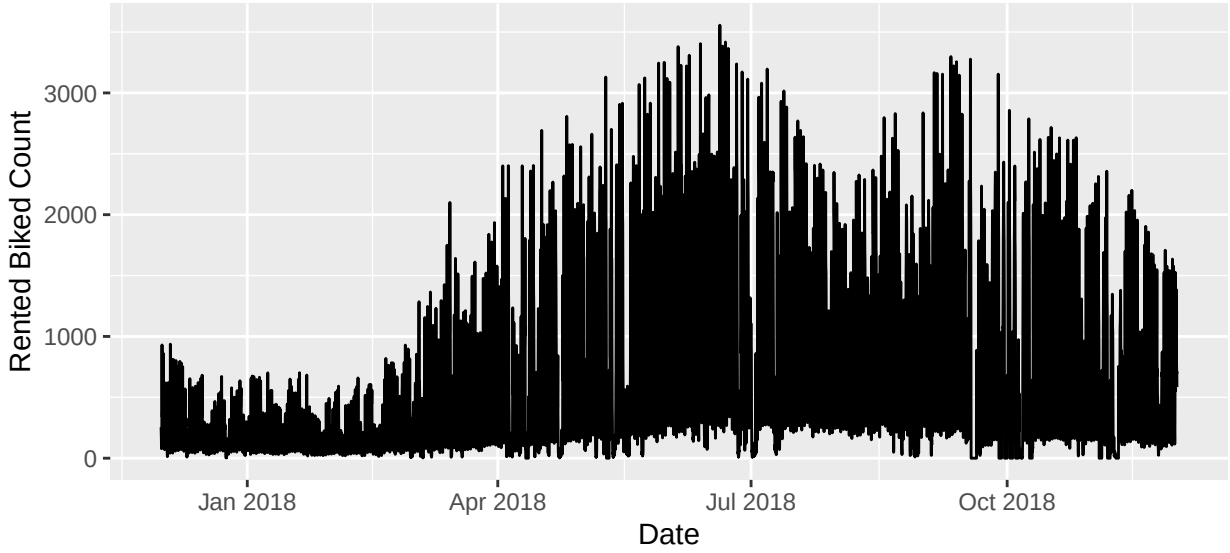

Figure 2.1: Hourly Rented Bike Count Over Entire Time Period

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).

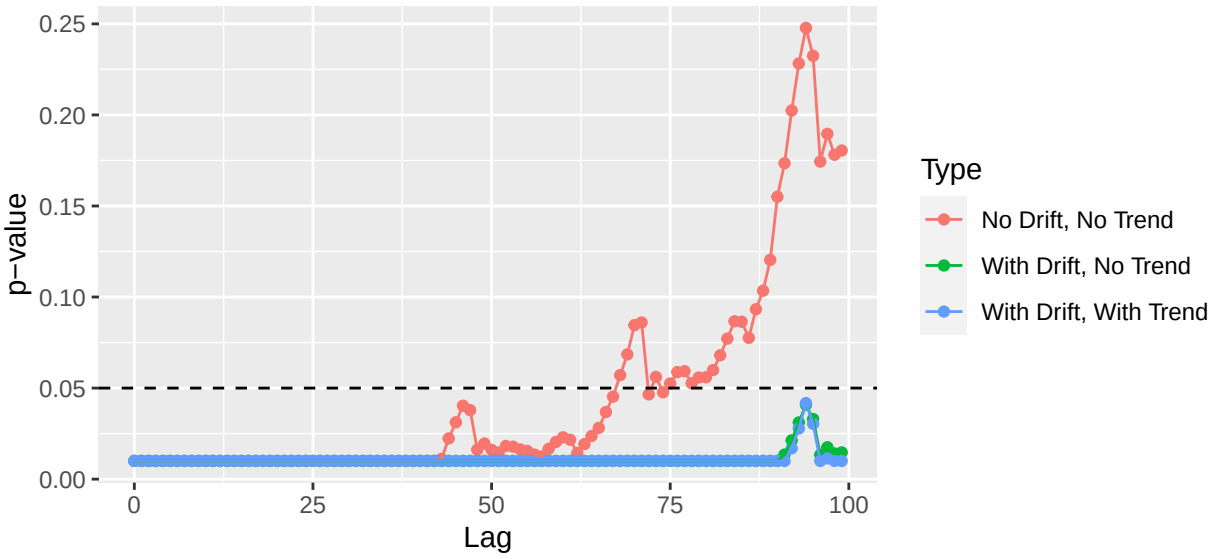

Figure 2.2: Augmented Dickey Fuller (ADF) Test for Stationarity

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.

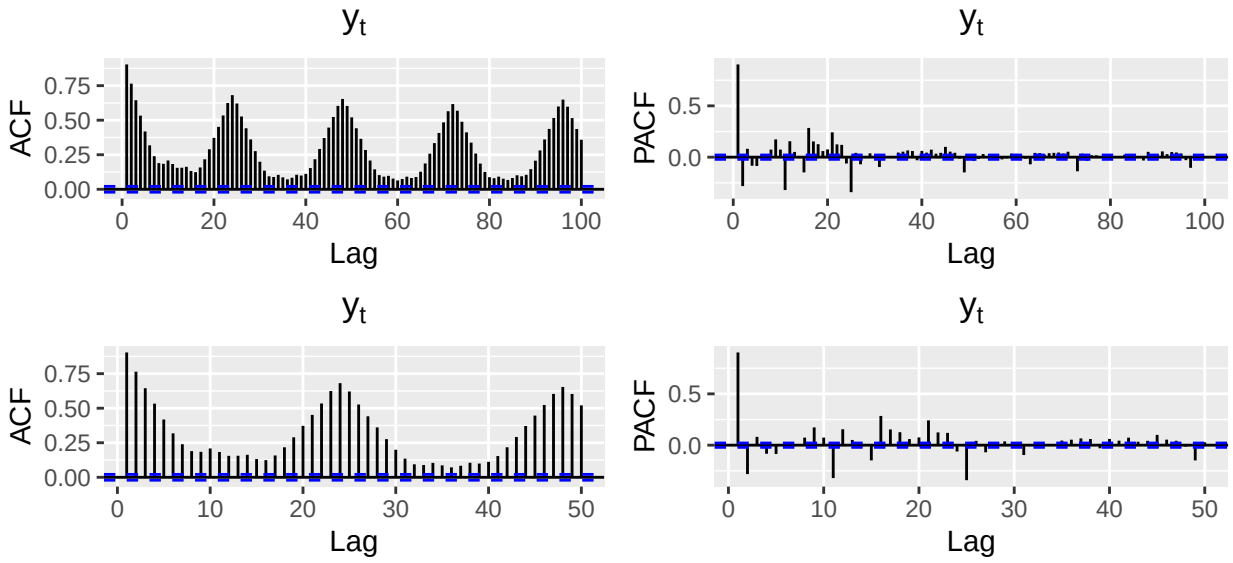

Figure 2.3: Autocorrelation and Partial Autocorrelation of Rented Bike Count, i.e., y_t

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 $h > q$ [3]. 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) or MA(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.

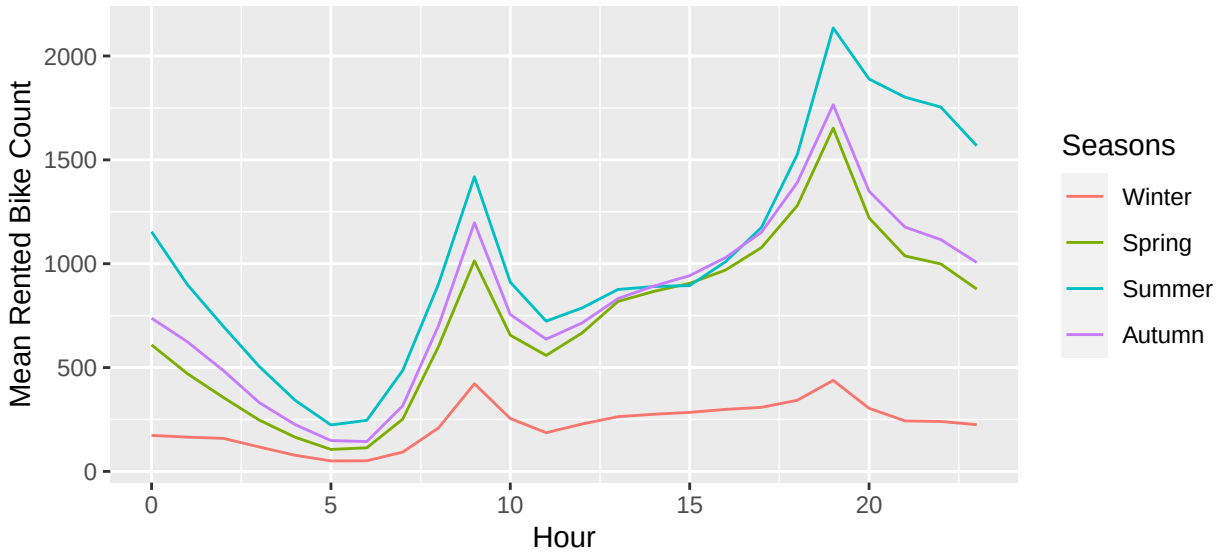

Figure 2.4: Rented Bike Count by Hour Grouped by Seasons

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).

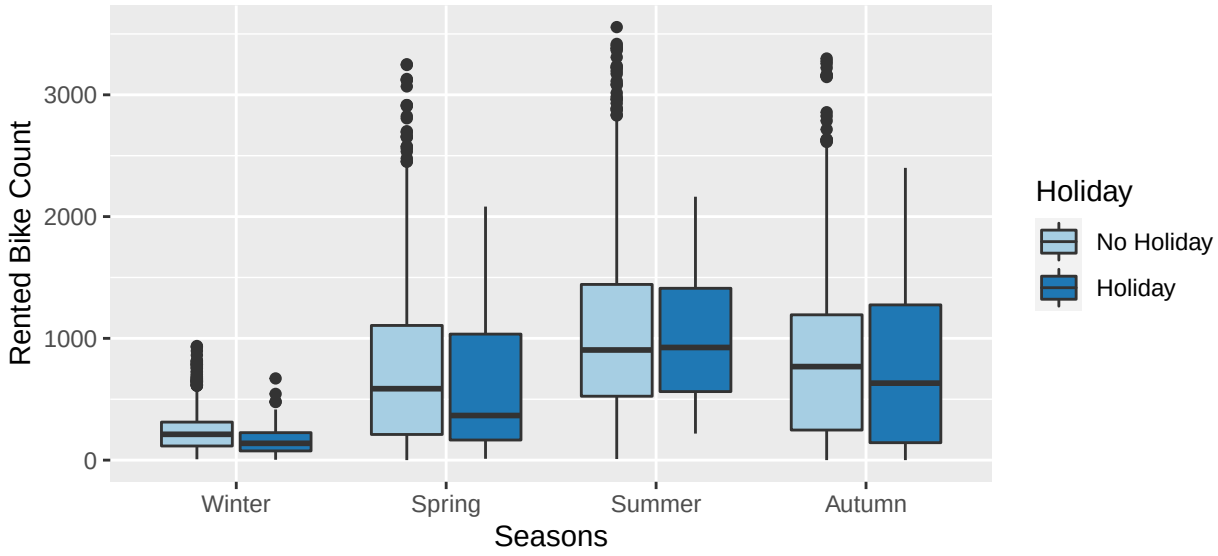

Figure 2.5: Rented Bike Count by Season Grouped by Holiday

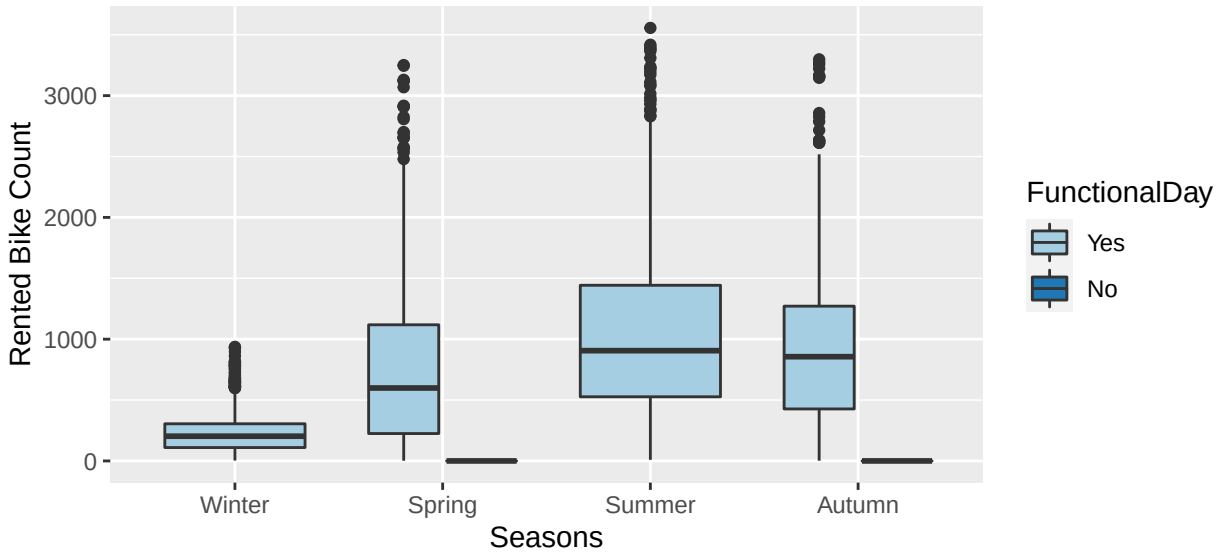

Figure 2.6: Rented Bike Count by Season Grouped by Functional Day

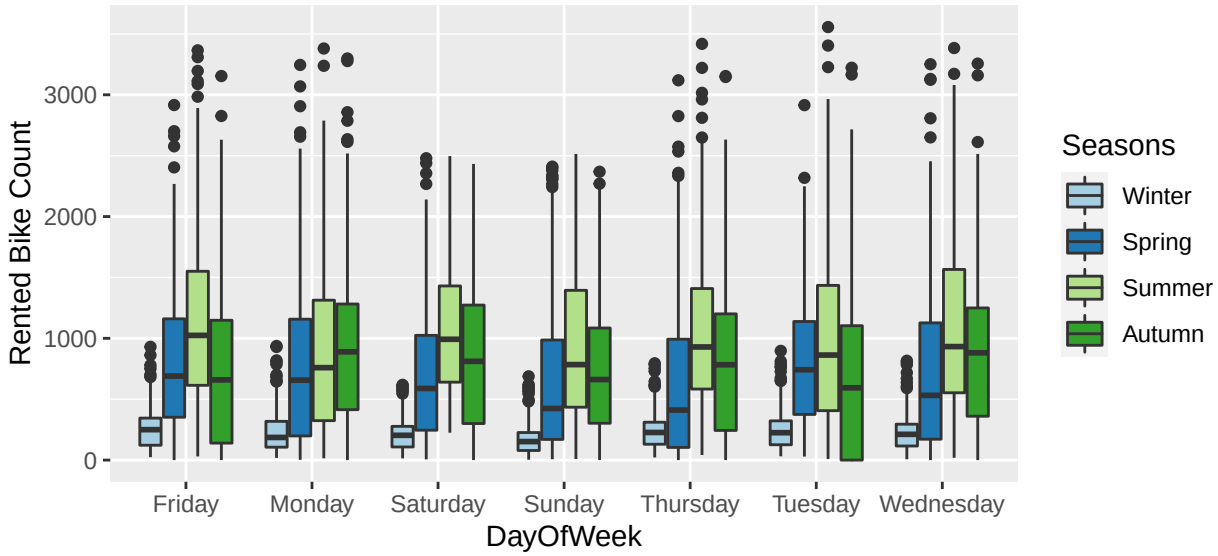

Figure 2.7: Rented Bike Count by Day of Week Grouped by Season

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**

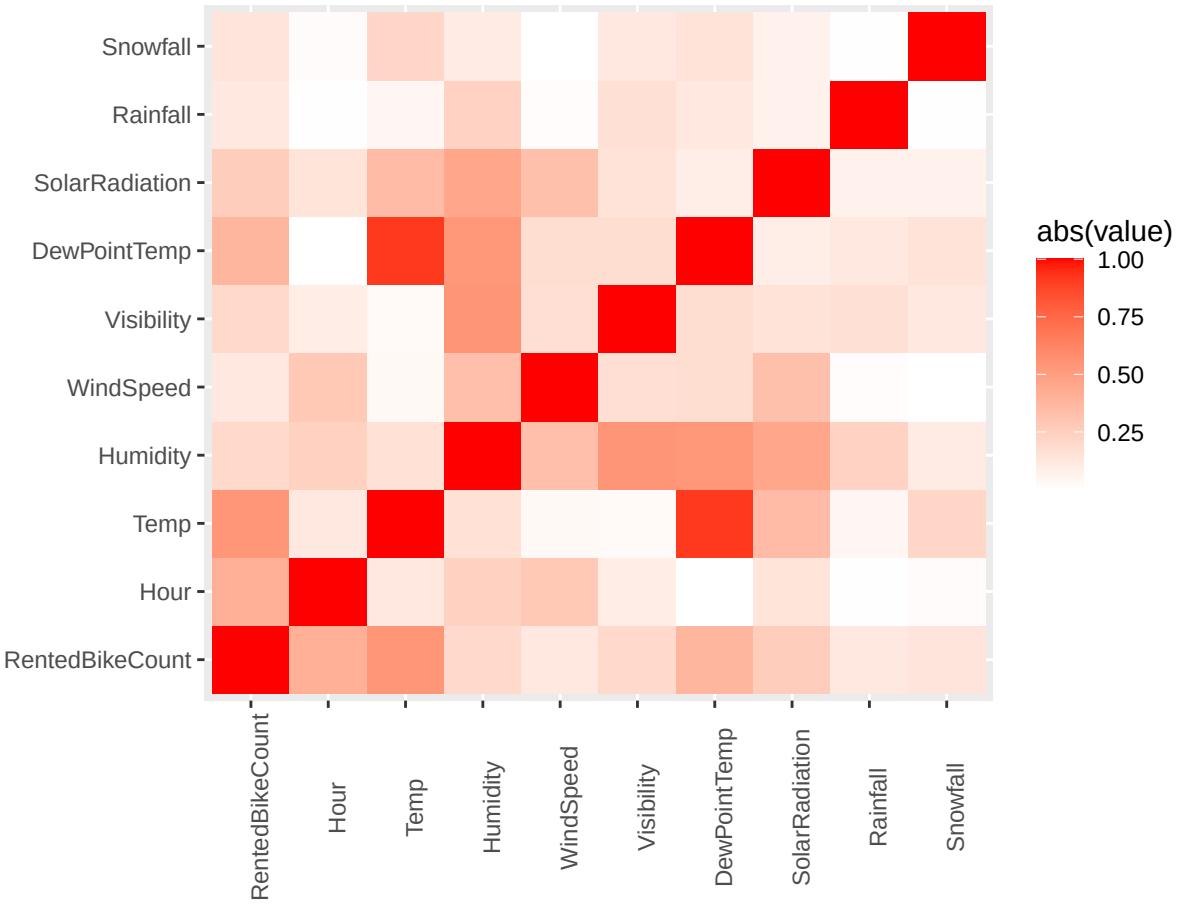

Figure 2.8: Covariates Correlation Matrix

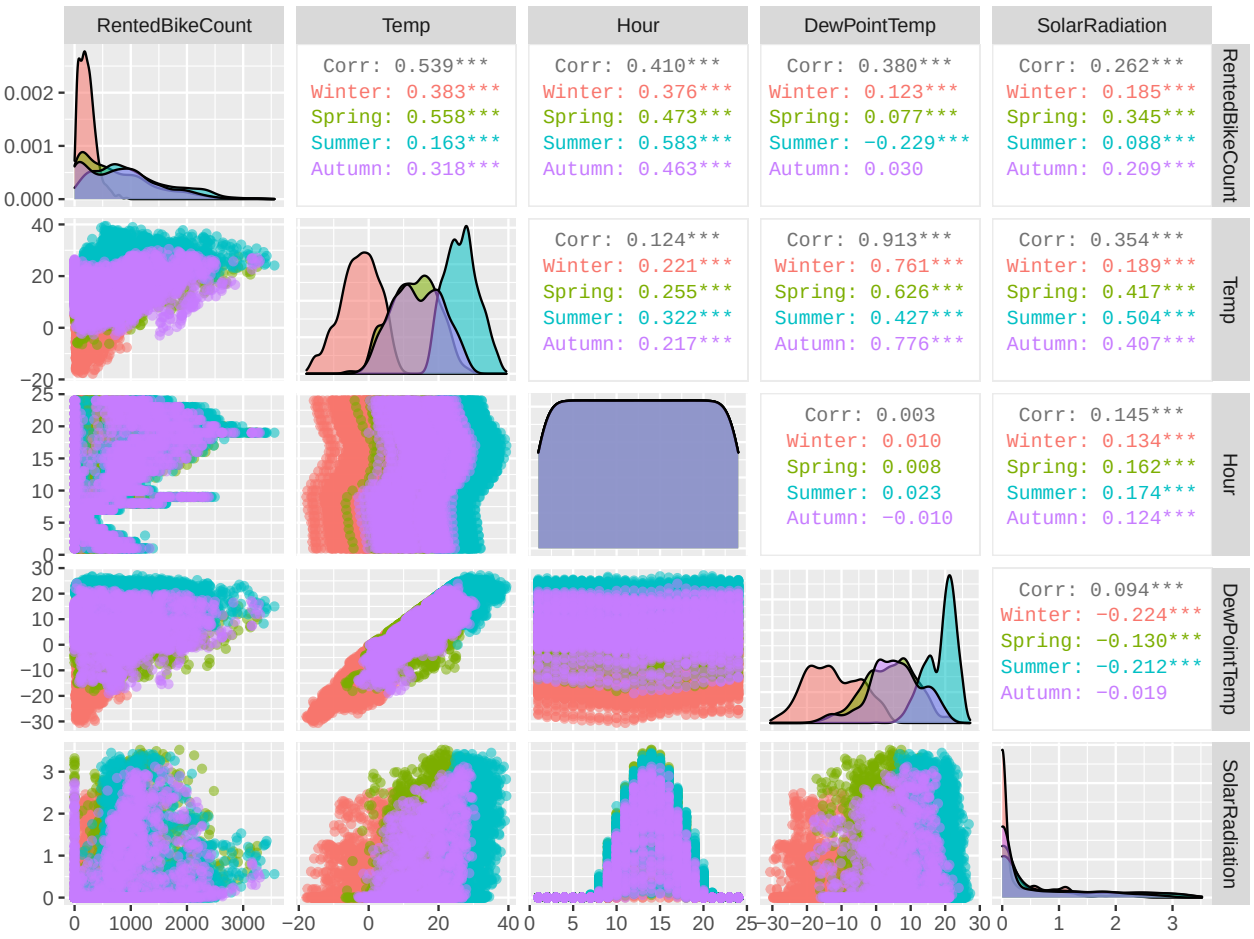

Figure 2.9: Top Four Covariates Correlation Matrix By Season

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.

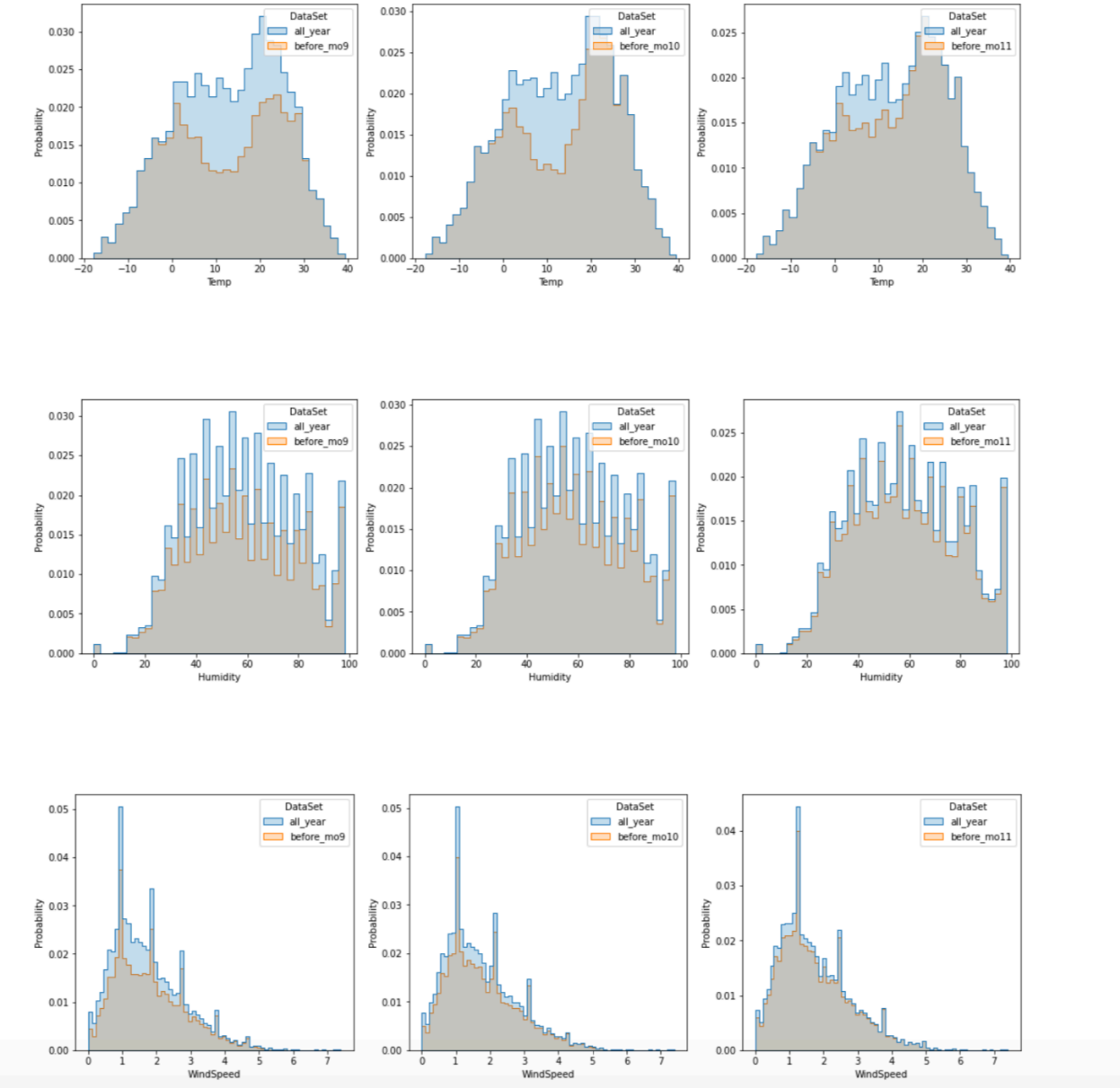

Figure 2.10 Temp, Humidity and WindSpeed Distribution

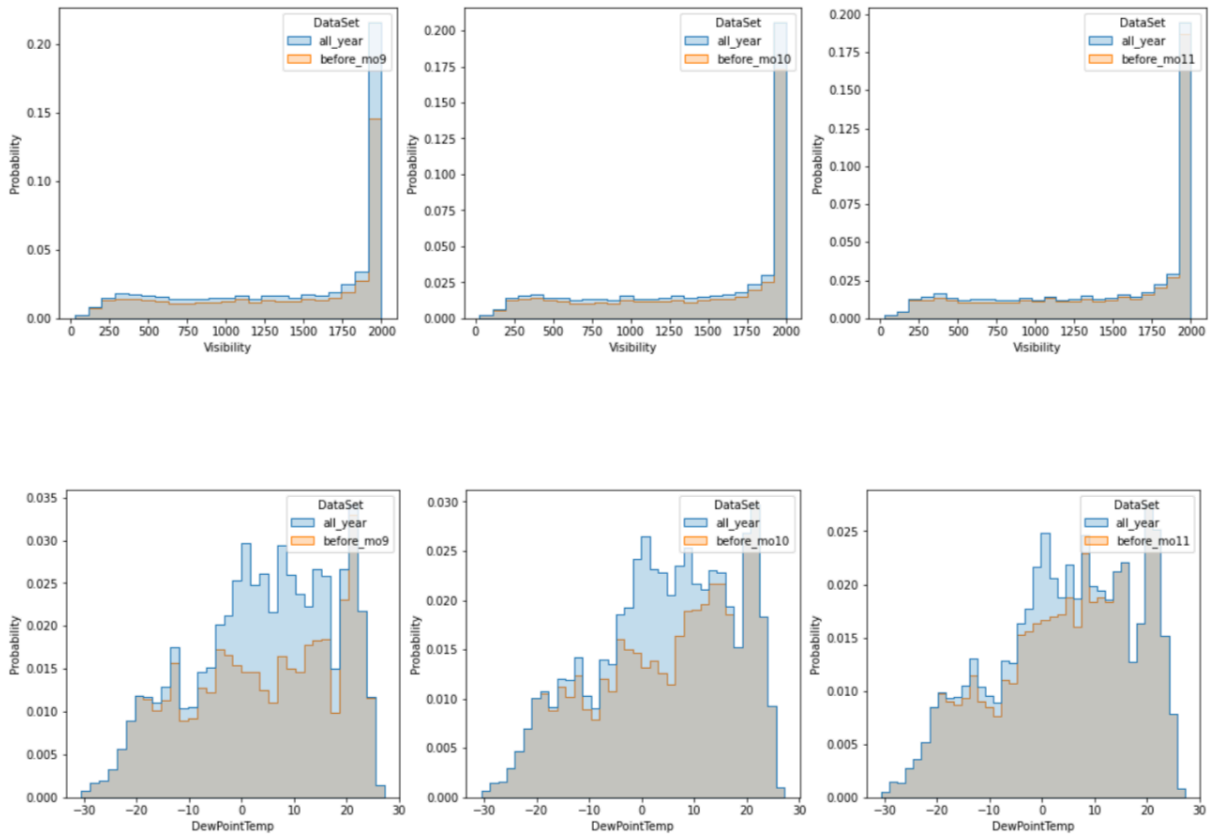

Figure 2.11 Visibility and DewPointTemp Distribution

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 $\widehat{f} : \mathbb{R}^p \mapsto \mathbb{R}$, such that we can make predictions with $\widehat{f}$, i.e., $\widehat{f}(X) = \widehat{y}$, where $x \in \mathbb{R}^p$ and $\widehat{y} \in \mathbb{R}$, such that $\widehat{y}$ is a "good" prediction of y, the unobserved supervisor.

One way to define "good" is to define it in the context of "loss", specifically $\ell(\widehat{y}, y)$. There are a multitude of loss functions to consider, but a "popular" loss for regression is the squared error loss, i.e., $\ell(\widehat{y}, y) = (\widehat{y} - y)^2$, 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, $\widehat{\beta} = \arg\min_\beta ||y - X\beta||_2^2$.

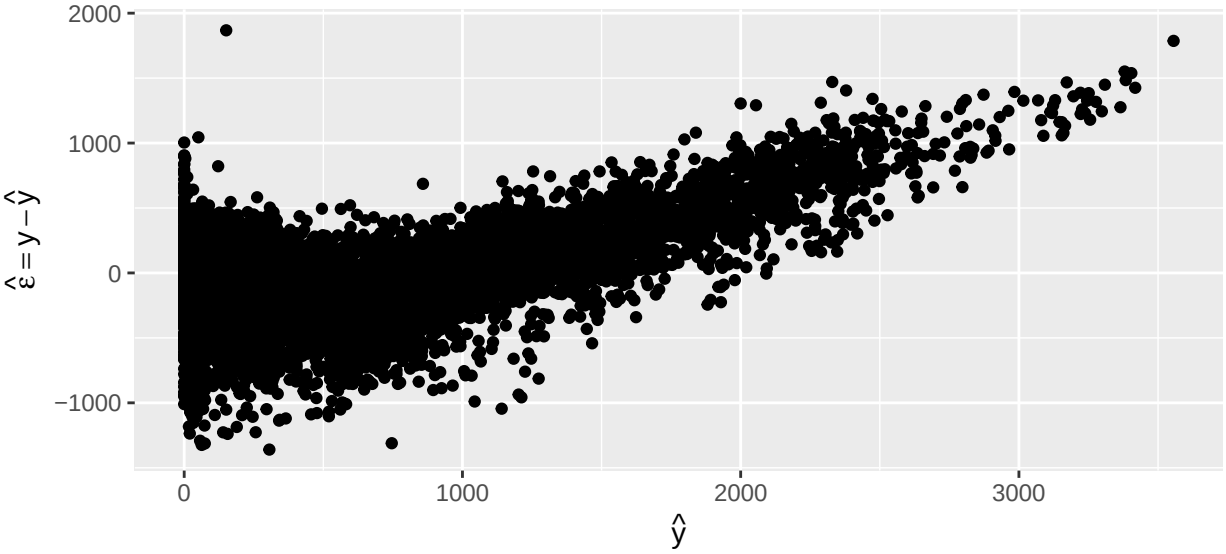

Figure 3.1: Residual Plot for Linear Regression

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 $\hat{y}$ 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, $\hat{\beta}(\lambda_1, \lambda_2) = \arg\min_\beta ||y - X\beta||_2^2 + \lambda_1||\beta||_1 + \lambda_2||\beta||_2^2$.

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, $\hat{\beta}(\lambda, \alpha) = \arg\min_\beta (||y - X\beta||_2^2 + \lambda((1-\alpha)||\beta||_2^2 + \alpha||\beta||_1))$. 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., $\sum_{i=1}^n (y_i) - \hat{y}_i)^2$. 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.

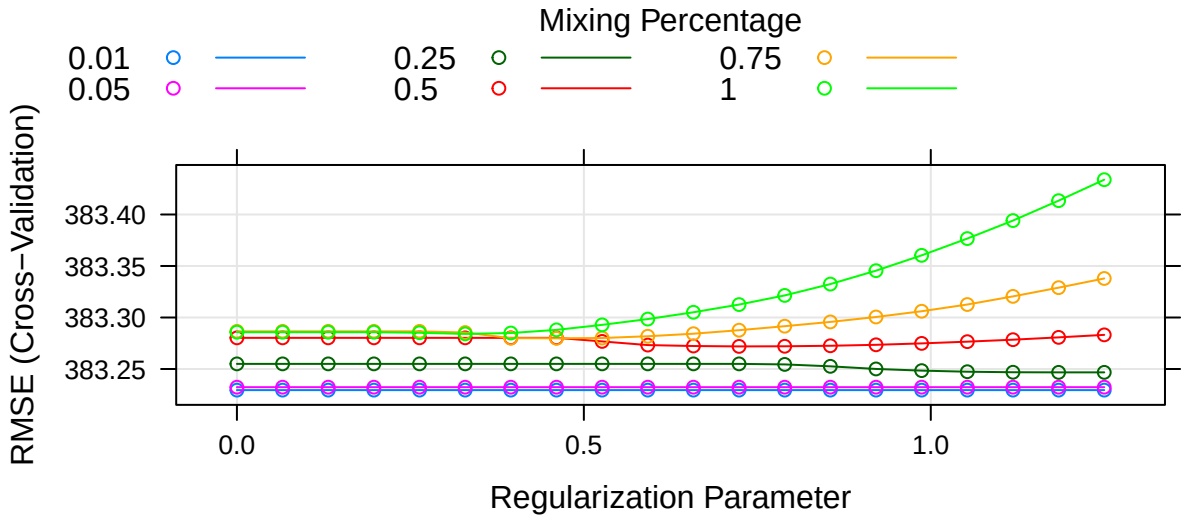

Figure 3.2: Cross-Validation Risk Estimates for Elastic Net Hyper Parameters

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.

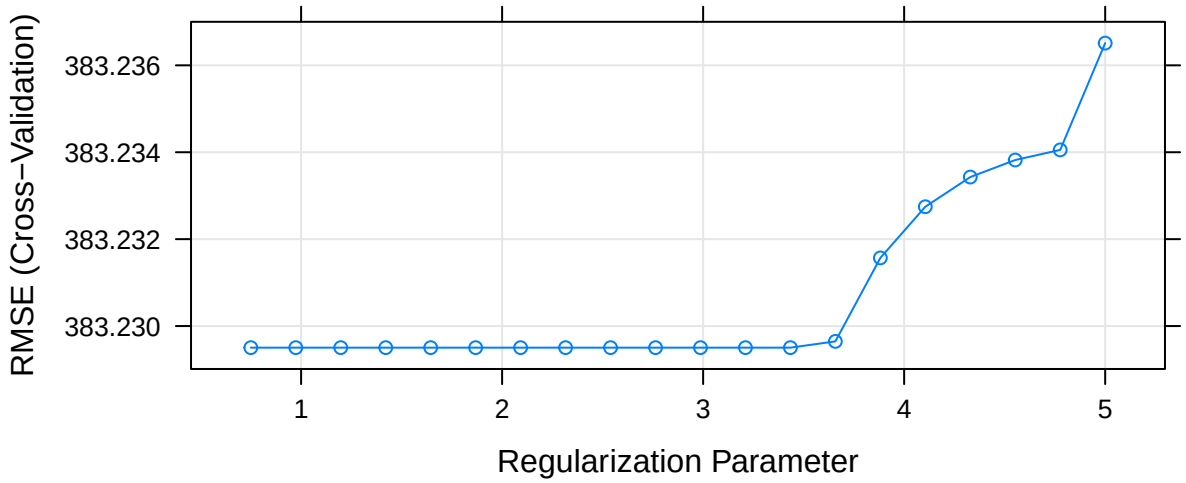

Figure 3.3: Cross-Validation Risk Estimates for Elastic Net Hyper Parameters Past Boundary Condition

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

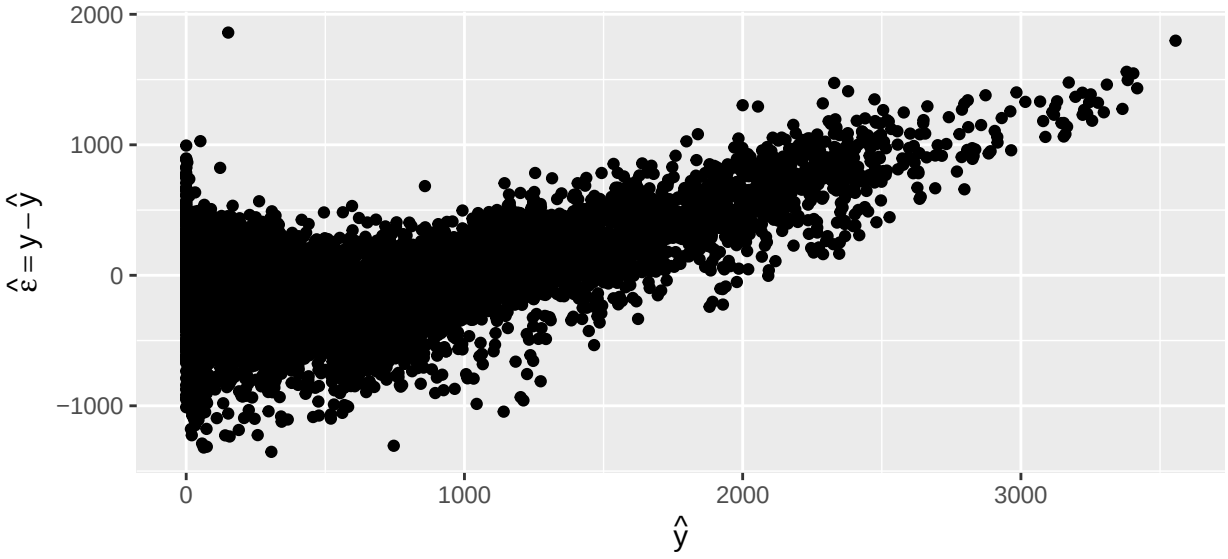

Figure 3.4: Residual Plot for Elastic Net

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.

3.3 Non-Linear Methods

3.3.1 Multivariate Adaptive Regression Splines (MARS)

Multivariate Adaptive Regression Splines (MARS), uses the linear regression framework but constructs features to model non-linearities and their interactions in an automated fashion. Specifically, in a foward-stepwise fashion, it looks over all p features and a specified set of "knots" to identify the most relevant hinge feature to introduce, e.g., $)I(x_j - \kappa_0 > 0$ and $I(x_j - \kappa_0 \leq 0)$. Further, it can introduce interactive features, e.g., $I(x_j - \kappa_0 > 0) \times I(x_k - \kappa_1 \leq 0)$. MARS then uses generalized cross-validation (GCV) to determine some ideal subset of features.

As such, the number of knots (hinge points) to consider as well as the degree of interaction are hyperparameters.

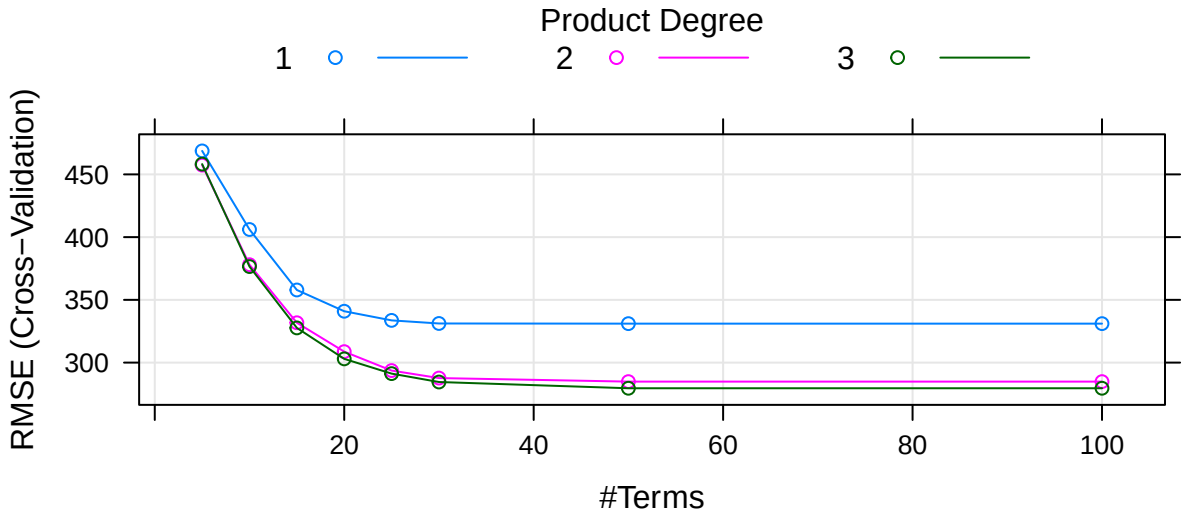

Figure 3.5: Cross-Validation Risk Estimates for MARS Parameters

Here, we can see that as we increase the number of knots, the cross-validation risk estimates lower, eventually leveling off after 50 knots. Further, having interactions help, but seemingly two-way interactions are enough and three-way interactions don't buy us much.

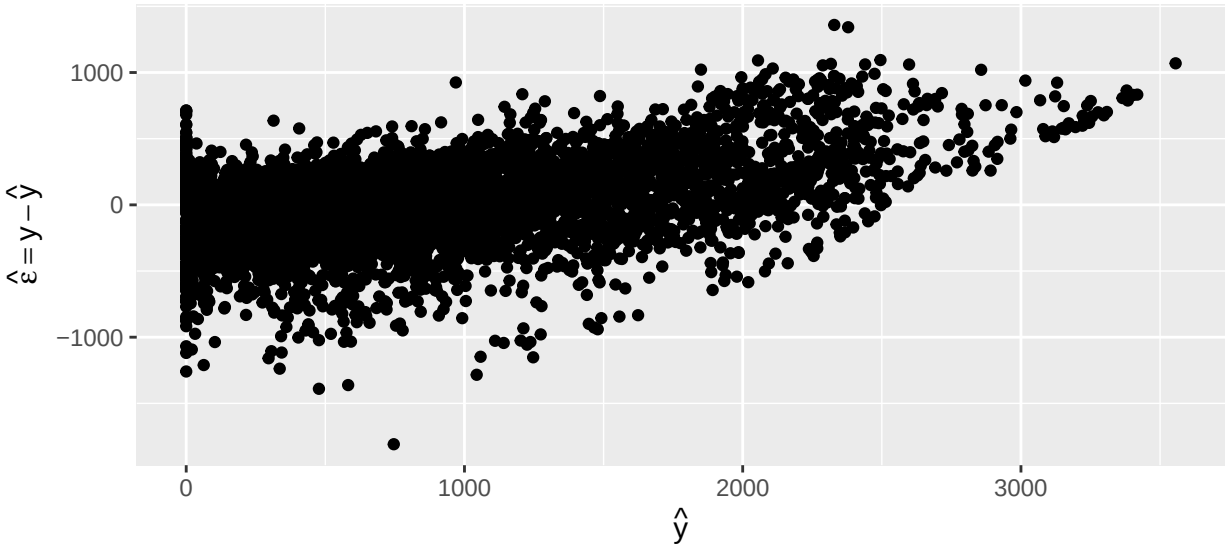

Figure 3.6: Residual Plot for MARS

Having introduced non-linearities, we're still using effectively a linear regression framework and as such, we can talk about residual plots. In this case, while there's still not random error and a clear trend, it's not as pronounced as the previous strictly linear predictors.

3.3.2 Decision Tree

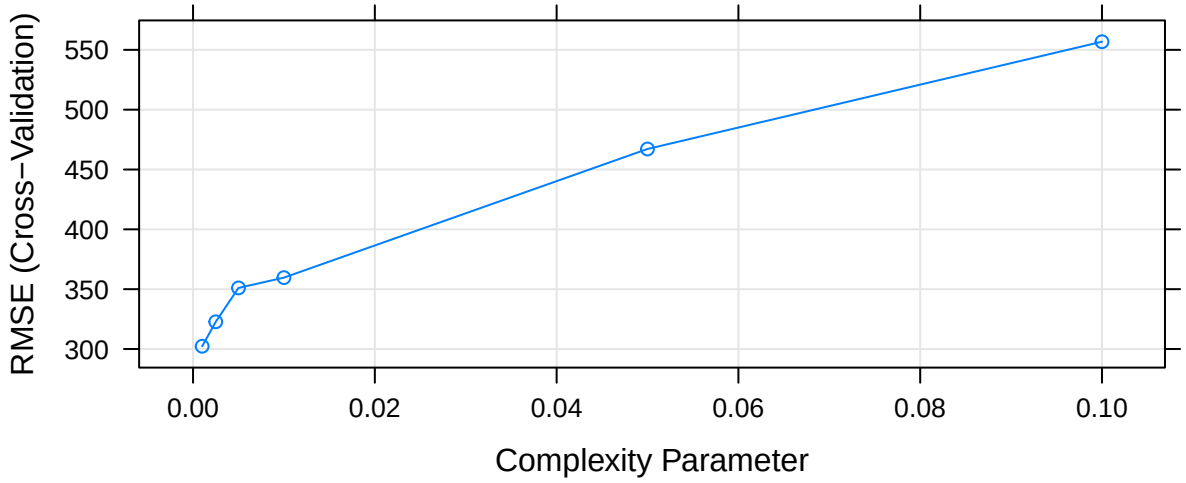

Figure 3.7: Cross-Validation Risk Estimates for Decision Tree Parameters

The only hyperparmaeter of interest is "Complexity Parameter," which represents the minimum amount of improvement needed from a split in order for the split to be considered. As such, as we increase this hyperparmeter, the more pruned our tree becomes, i.e., it's not as deep. Here, we can see that as we increase this hyperparameter, the cross-validation risk-estimate is increasing. This suggests that our best tree is a fully growned decision tree and that any pruning would be detrimental.

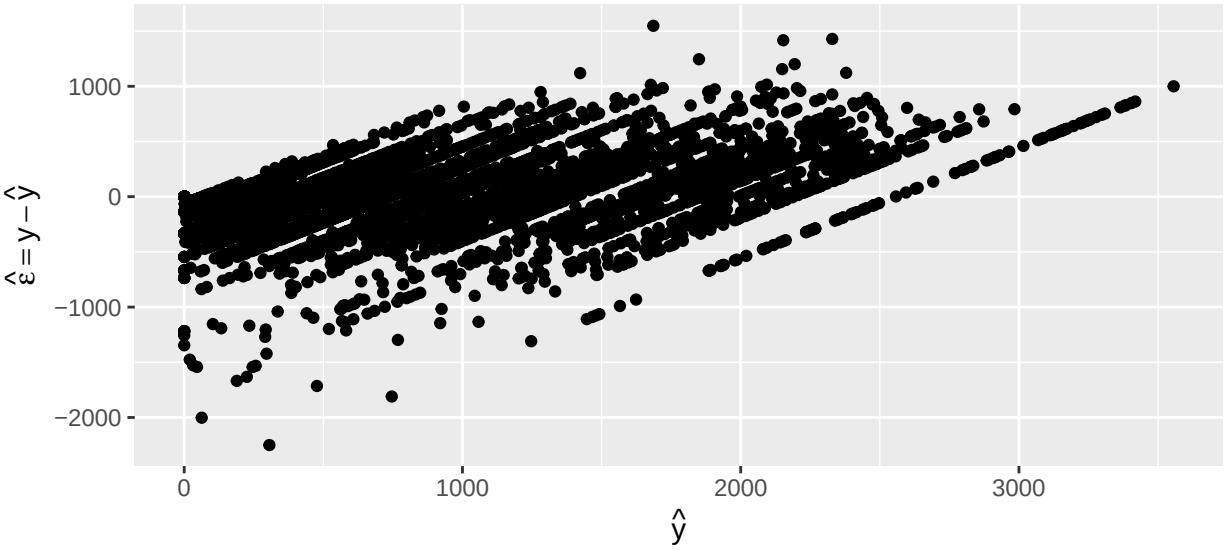

Figure 3.8: Residual Plot for Decision Tree

At this point, it may not be meaningful to do in-sample residual plots due to the induced non-

linearities.

3.3.3 Random Forest

Decision trees are perceived as low bias high variance procedures. This is to say when we permute the data set, decision trees don't necessarily generalize well. As such, we want to maintain the low bias while also lowering the variance. Bagging is an approach of using an ensemble of trees, where for each tree we bootstrap our data and the tree is fit upon that bootstrap data. The concern is that we might not get as much variance decrease as we'd like, since for a particulraly strong covariate, it might show up in every single tree, thus inducing dependency therefore limiting the variance decrease. Random Forest builds upon bagging by for each split, we only determine a subset of the covariates, typically $\sqrt{p}$, where p is the number of covariates.

3.3.4 Boosting

Chapter 4

Dependency of Y_t

4.1 Autocorrelation and Partial Autocorrelation of Rented Biked Count, y_t

In section 2.2, the ACF and PACF plots are suggesting strong autocorrelation of the past bike demand. The ACF shows a clear daily trend (every 24 lags). The hour feature used in estimators can be used to reflect this daily trend. The PACF shows significant dependence between demand and past demands - lag 1, 2, 3, 4, 5, 8, 9 ,10, etc. However, this past demand information is not well used in previous estimators. A simple way to use the information is to add dependency features of past demand.

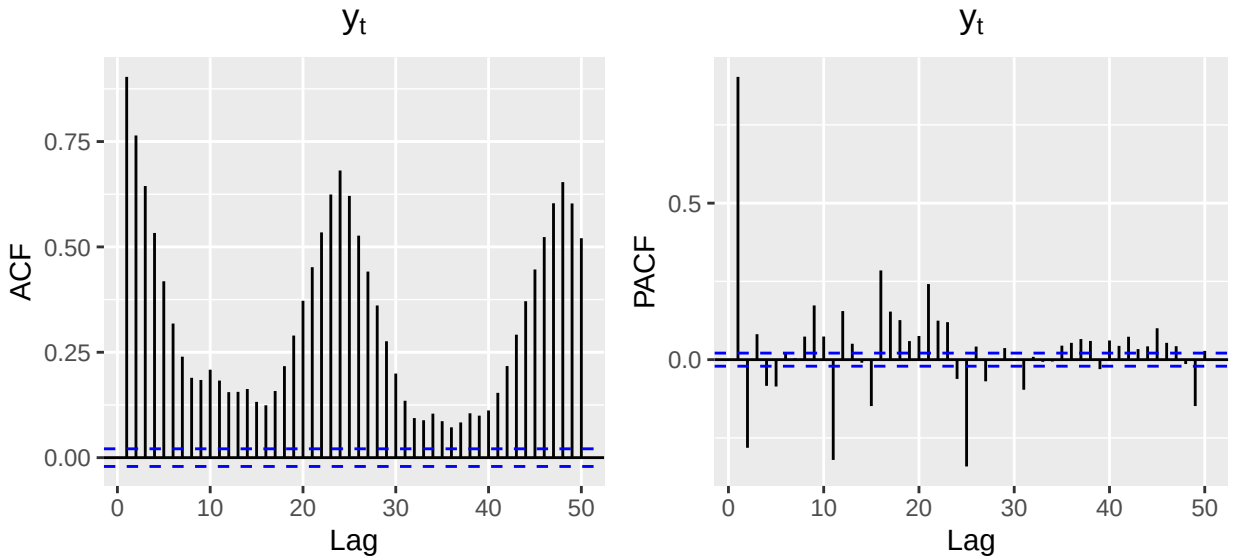

Figure 4.1: Autocorrelation and Partial Autocorrelation of Rented Bike Count, i.e., y_t

4.2 Dependency Feature

In addition to weather feature and hour feature, add dependency features for past demand: e.g., lag_1 means demand in the last hour; lag_4 means hourly demand 4 hours ago, etc. By adding dependency features, the top observations will have missing value as past demand information is not available. The obaservation with missing values will be deleted from training data.

4.2.1 Estimation without Dependency Features

In previous study, boosting method brings the best result among all estimation methods. Therefore, the estimator using boosting method with parameter tuned from previous study will be used in all following model training and predictions. All observations before anchor date Nov 1, 2018 will are used for training and the 30 observations from Nov 1, 2018 to Nov 30, 2018 are used for testing. To compare the impact of dependency feature, model M_0 is fit with only weather features and hour information. The test R^2 in M_0 is 0.756. The figure below is showing the feature importance from M_0, which suggesting 'Temp' as the most important features effecting the demand.

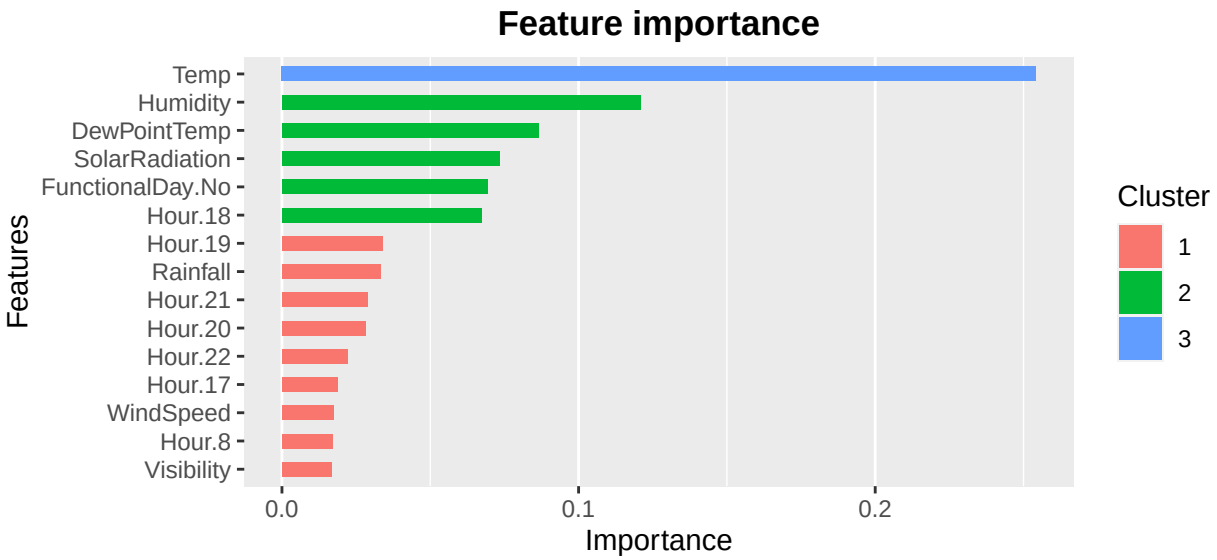

Figure 4.2: Feature Importance M_0

4.2.2 Estimation with Dependency Features

Based on PACF plot, dependency features within 1 day (24 hours) are selected for the model fitting, including the past 1-24 hour demand except for lag 6, 7, 14, as M_1. With the additional 21 columns

added to X, the test R^2 goes up to 0.967 - a significant improvement from M_0. In addition, the feature importance plot for M_1 below shows the past demand features are more important than weather features compared with M_0.

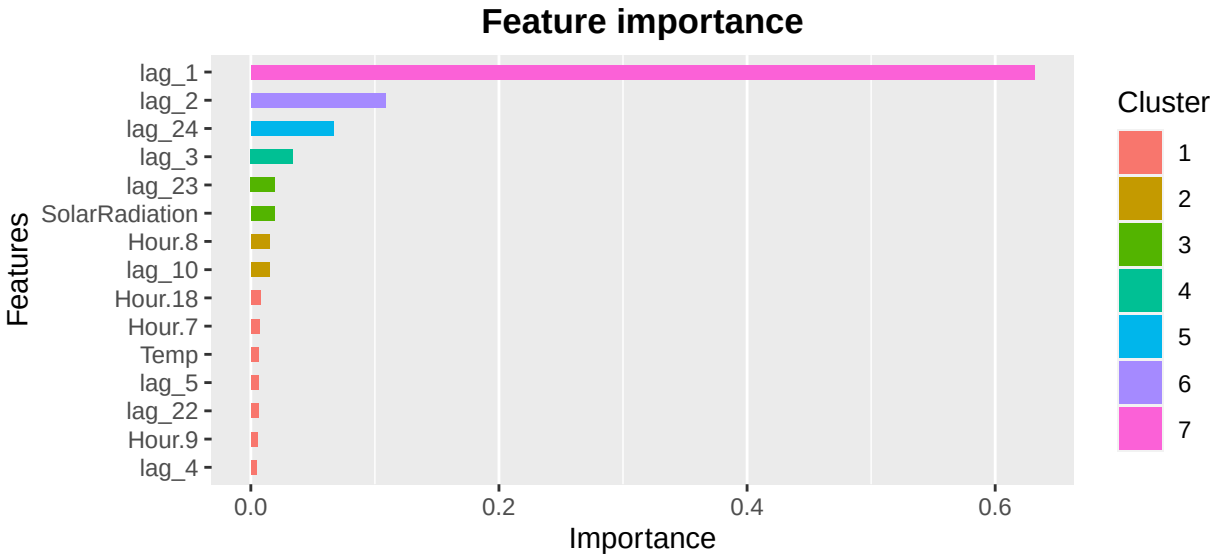

Figure 4.3: Feature Importance M_1

4.2.3 Estimation with Reduced Dependency Features

Based on the feature importance plot of M_1 and the PACF plot, the dependency features can be reset to lag 1, 2, 11, 16 and 24 to reduce the size of X variables. Thus M_2 is fitted with smaller number of dependency features and the test R^2 is 0.958, which is less than 1% lower than M_1 and significant higher than M_0.

4.2.4 Estimation with Dependency Features Defined by Business Assumptions

In M_3, the dependency features added is lag 1, 2, 24, 48 and 24×7 based on business assumptions that the demand is related to the past demands 1 and 2 hours ago, 1 day ago and 1 week ago. The test R^2 is 0.95, which is as good as M_1 and M_2.

Table 4.1: Test R2 Comparison of Models with Different Dependency Features

	Model	TestR2
No_Dependency_Feature	M0	0.756
Full_Dependency_Feature	M1	0.967
Reduced_Dependency_Feature	M2	0.958
Business_Dependency_Feature	M3	0.950

4.2.5 Estimation Comparison

The table below is comparing the test R^2 among models with different dependency features. Considering the test accuracy and variable size, M_2 and M_3 are preferred.

Chapter 5

Forecasting Application

5.1 Business Scenaros

The purpose to predict bike demand is to make bikes available and accessible to the public at the right time. Thus, the forecasting of hourly bike demand is required to support business decisions and operations. Based on system infrastructure capacity, we define two typical business scenarios below in real application.

5.1.1 Daily Data Update

The system data is updated on a daily base for further analysis and forecasting to the next 24 hours' **hourly demand** is required for high-level planning of the next day. To test the performance in this business scenario, we assume the data is updated at 0:00 AM of the day and all available data at the moment is used for model training to predict the next 24 hour demands.

5.1.2 Real-time Data Update

If the system infrastructure could support real-time data update, an hourly model training could be run to predict the next **hour demand**. Any changes in the past hours could be used in the next hour demand prediction. To test the performance, the models will be trained hourly with all the data available at the moment (include demand data from last hour) and used to predict the demand in the next coming hour.

Table 5.1: Foresting Result with Daily Data Update and Dependency

Train_R2	Train_CV_R2	Test_R2
0.977	0.799	0.86

5.2 Hourly Demand Forecasting with Daily Data Update

5.2.1 Estimator and Dependency Features

When data is updated once every day, the latest observation available to use as dependency feature is the lag 24 for all prediction time stamps. Therefore, the smallest lag of dependency features can be added is lag 24. Based on the business assumption in dependency feature study, lag 24, 48 and 24×7 are added as dependency features, which represent the past demand from same hour 1 day ago, 2 days ago and 1 week ago. The model training is repeated daily for November 2018. And the forecasting method is boosting using the parameters tuned from previous study.

5.2.2 Forecasting Results

In a daily data update, there are 30 repeated model training and forecasting (1 in each day). In each iteration of model training and forecasting, all observations prior to the iterator date are used as training data and the next 24 hours' hourly demands are used as testing data. The train R^2, mean CV R^2 and test R^2 results are recorded in each iteration.

The table below (see Table 5.1) shows the average training R^2 over the 30 iterations is 97.7% and the average mean CV R^2 over the 30 iterations is 79.9%. There's no large discrepancy between the cross validation R^2 and test R^2. Therefore, there's no strong evident of overfitting issue.

With daily data update, some poor prediction exist in certain iterations, which is shown clearly in the figure below comparing forecast demand and real demand (See Figure 5.1): on Nov 3, Nov 6 and Nov 9, when there's no demand due to non-functional day, the forecasting at the beginning of the day still predicts certain amount of demand.

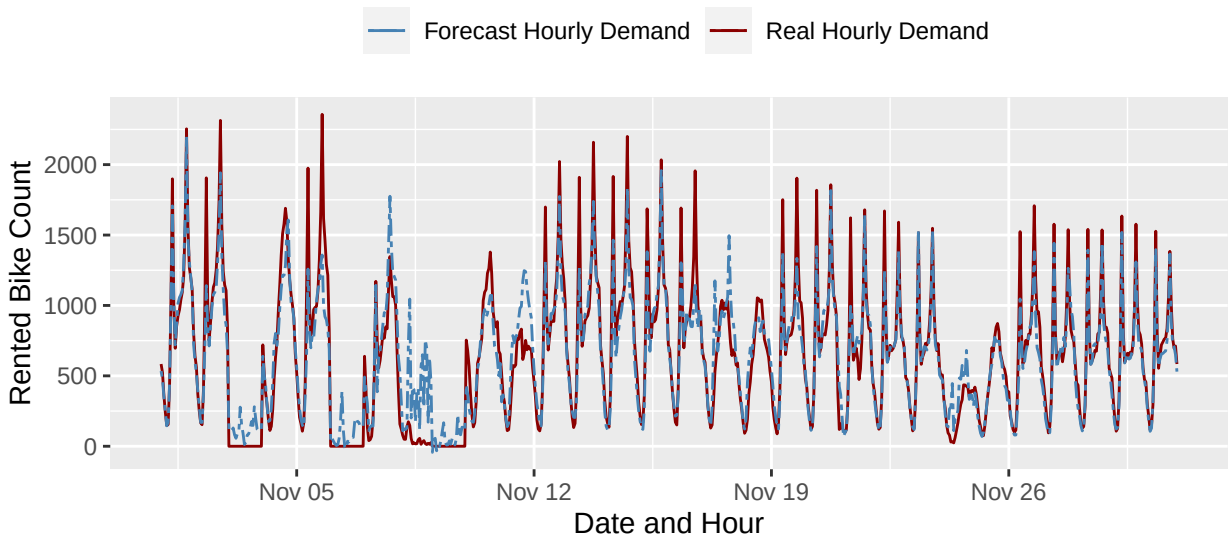

Figure 5.1: Forecasted Demand and Real Demand Comparison with Daily Data Update and Dependency

5.3 Hourly Demand Forecasting with Real-time Data Update

5.3.1 Dependency Features

When the data is updated in real-time, all past demands up to one hour ago (lag 1) can be used as dependency feature. Therefore, we add the past demand from 1 hour ago, 2 hours ago, 1 day ago and 1 week ago as dependency features to the data (M_3 in previous study). The modeling training is repeated every hour and used to predict demand only for the coming hour. And the forecasting method is boosting using the parameters tuned from previous study.

5.3.2 Forecasting Results

In the real-time data update, there are 30×24 repeated model training and forecasting (1 in each hour). In each iteration of model training and forecasting, all observations prior to the iterator date and hour are used as training data and the next 1 hour demands is used as testing data.

The table below (see Table 5.2) shows both train R^2 and average mean CV R^2 over the 30×24 iterations are above 90%. The test R^2 reaches 96%. And the figure comparing forecast demand and real demand below (See Figure 5.2) shows a very good match between the two curves, representing an accurate prediction. Moreover, with the real-time data update, the system is able to know the

Table 5.2: Foresting Result with Real-time Data Update and Dependency

Train_R2	Train_CV_R2	Test_R2
0.995	0.929	0.964

Table 5.3: Forecasting Results Comparison with Dependency

	DailyDataUpdate	Real.timeDataUpdate
Train_R2	0.977	0.995
Train_CV_R2	0.799	0.929
Test_R2	0.860	0.964

latest demand in the past hour and adjust the coming hour demand prediction - forecast demand in Nov 3, Nov 6 and Nov 9 stays low when detecting low demand in the previous hours.

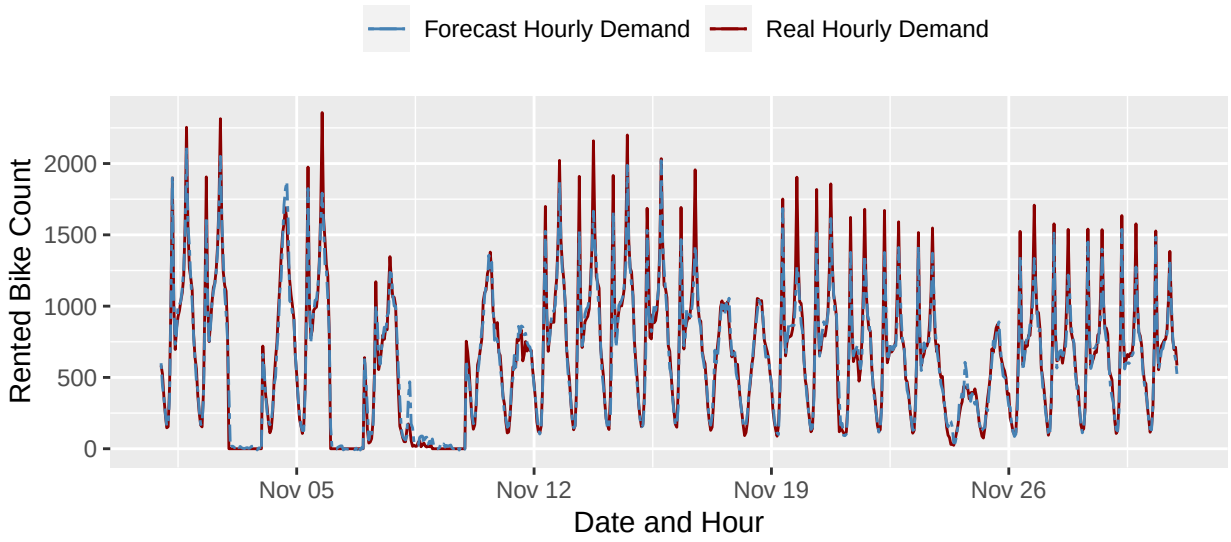

Figure 5.2: Forecasted Demand and Real Demand Comparison with Real-time Data Update and Dependency

5.4 Forecasting Results Comparison

5.4.1 Improvement with Real-time Data Update

If the system could support real-time data update, the test R^2 shows a 10% improvement (see Table 5.3). Comparing the forecast demand to real demand, the real-time data update forecasting shows a much lower discrepancies than the daily data update forecasting (See Figure 5.3 and 5.4).

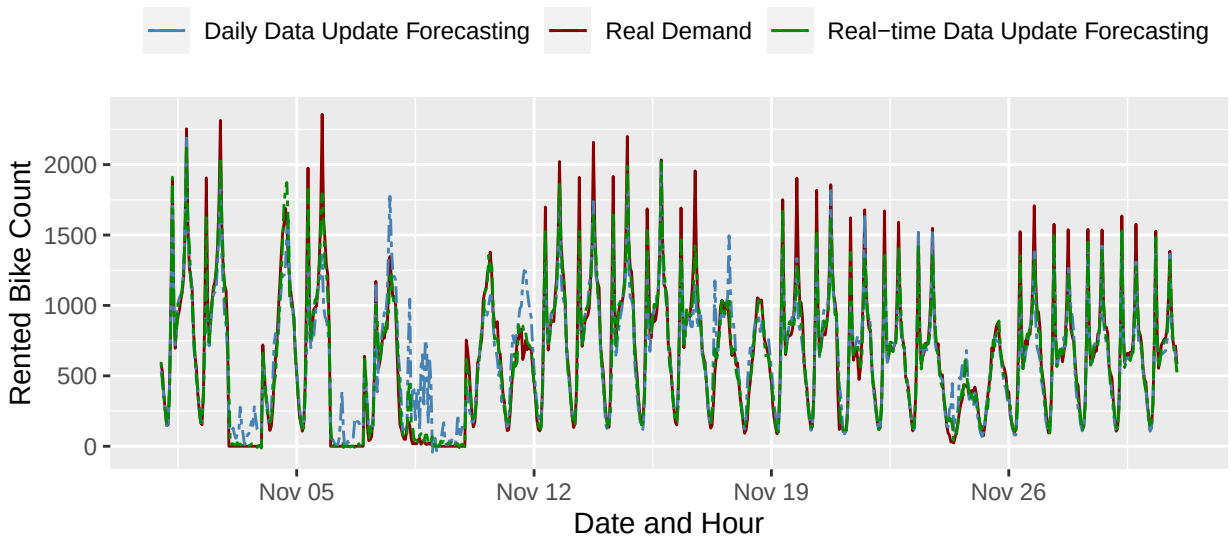

Figure 5.3: Forecasted Demand Comparison with Dependency

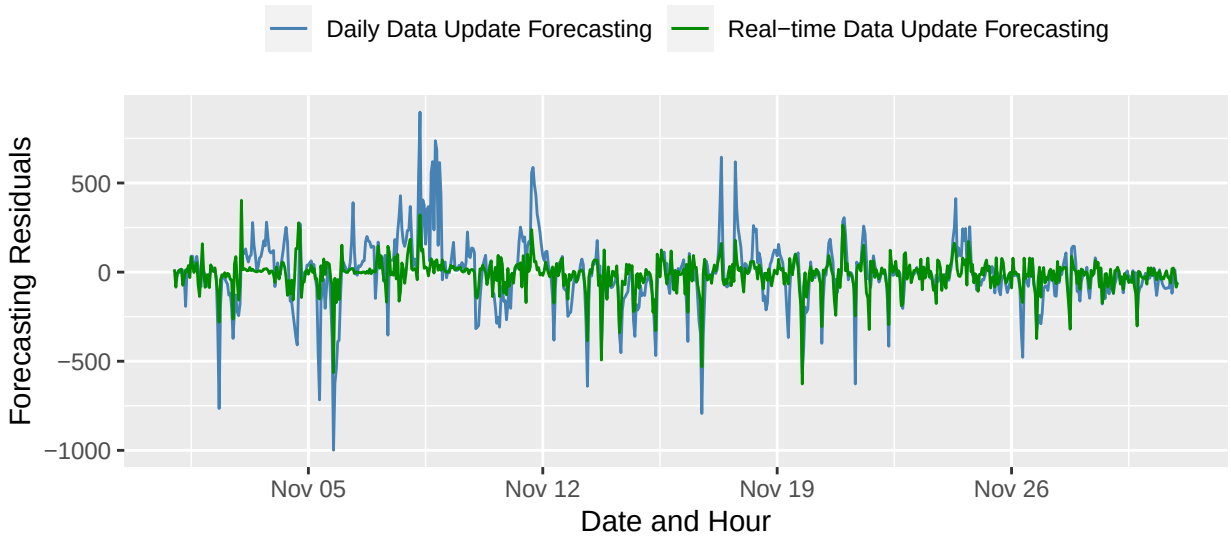

Figure 5.4: Forecasting Residual Comparison with Dependency

5.4.2 Improvement with Dependency

To better understand the importance of dependency, the same forecasting studies (repeated model training and forecasting on daily base and hourly base) are conducted without dependency features.

The table below (see Table 5.4) shows 8.5% improvement in daily data update and 14.7% improvement in real-time data update in terms of test R^2. Comparing the two figures plotting forecasting residuals (See Figure 5.5 and 5.6), there's more significant improvement by adding dependency

Table 5.4: Forecasting Results Comparison with and without Dependency

	Train_R2	Train_CV_R2	Test_R2
DailyUpdateWithDependency	0.977	0.799	0.860
RealTimeUpdateWithDependency	0.995	0.929	0.964
DailyUpdateNoDependency	0.960	0.686	0.775
RealTimeUpdateNoDependency	0.960	0.686	0.817

features with real-time data update than the scenario of daily data update - the blue residual line in the second figure is much more smooth than the green line compared with the first figure.

Moreover, if the system could not support a real-time data update, using the dependency features in the daily data update scenario still brings better (4.3% higher) forecasting accuracy than a real-time data update without dependency.

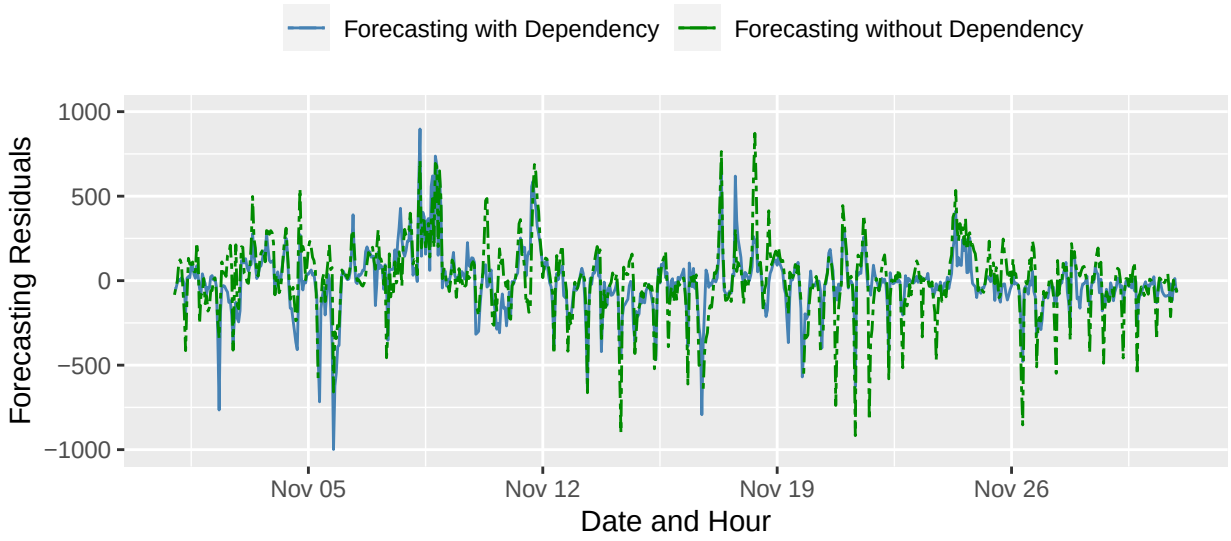

Figure 5.5: Forecasted Demand Comparison with Daily Data Update

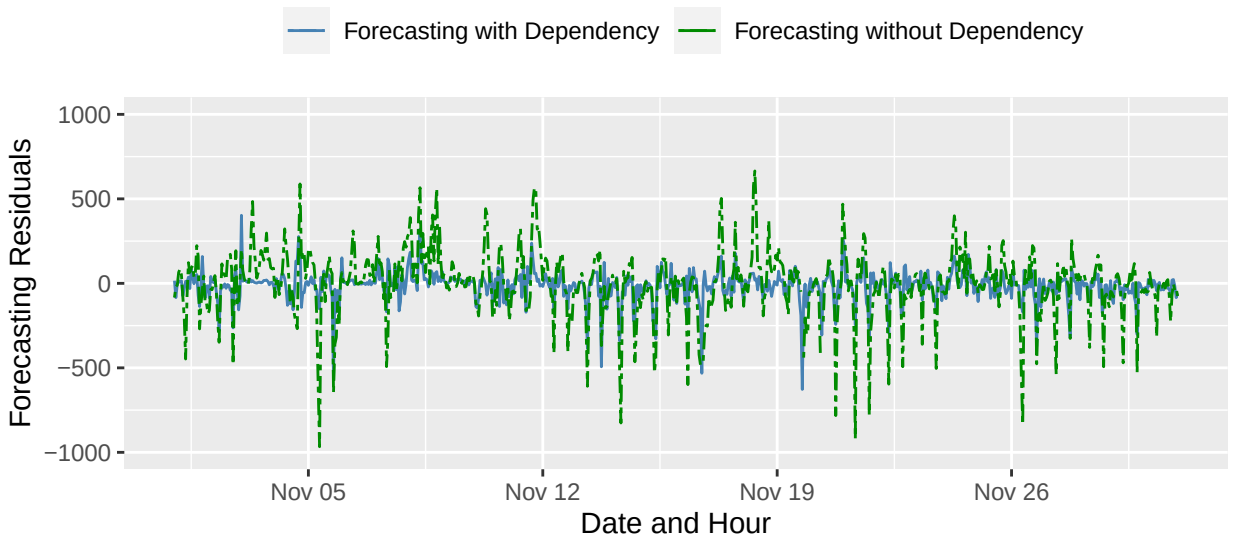

Figure 5.6: Forecasted Demand Comparison with Real-time Data Update

Chapter 6

Result and Conclusion

The study re-defines the training data set and testing data set by anchor date in a forecasting problem and re-evaluates estimation methods with the re-defined data sets. Among all estimation methods evaluated, Boosting brings the best result with lowest test error and highest test R^2.

Continuing with the Boosting estimator, the study incorporates the dependency of demand by adding additional predictor variables of the past hourly demand, which brings a significant improvement from previous estimation.

The estimation with Boosting method with selected dependency variables are applied to two business scenarios and the study approves 8.5% to 14.7% improvement in test R^2 by considering the dependency of past demand information. Meanwhile, the study also shows that the data infrastructure capacity affects the forecasting results - the more frequent data update, the more accessibility of latest bike demand and thus the more accurate prediction of the future.

Bibliography

[1] Sathishkumar V. E. and Yongyun Cho. “A rule-based model for Seoul Bike sharing demand prediction using weather data”. In: *European Journal of Remote Sensing* (Feb. 2020). An optional note, pp. 1–18.

[2] Sathishkumar V. E., Jangwoo Park, and Yongyun Cho. “Using data mining techniques for bike sharing demand prediction in metropolitan city”. In: *Computer Communications* 153 (Mar. 2020). Optional Note, pp. 353–366.

[3] Robert H. Shumway and David S. Stoffer. *Time Series: A Data Analysis Approach Using R*. Texts in Statistical Science. Chapman & Hall/CRC, 2019.