Modeling Seoul Bike Sharing Demand

Nam Tran, Bai Zou

November 27, 2020

Contents

List of Figures

List of Tables

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 uploaded to the UCI Machine Learning Repository (Dua and Graff 2017; E., Park, and Cho 2020; E. and Cho 2020). Further, the data contains 8760 observations, one supervisor, and 13 covariates.

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.

1.2 Previous Work

E., Park, and Cho (2020) 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, support vector machines with radial basis functions, as well as xgboost (an extension of gradient boosting machine). Ultimately, their best results were $R^2_{\text{train}} = 0.96$ and $R^2_{\text{test}} = 0.92$ using xgboost.

E. and Cho (2020) only varies marginally from E., Park, and Cho (2020) 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.

Our primary concern with previous work is that they don't make any reference to how they split up their 75% train and 25% test, e.g., is it randomly chosen interleaved train and test or is it a specific calendar day and everything after is test and everything prior is train? If interleaved, then the 75% training data's distribution and 25% test data's distribution are effectively identical and learning

on the train portion can be deemed as data leakage, since data has leaked from the test portion indirectly.

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

Initially, we consider two business requirements:

1. Predict next day hourly demand based on historical data up to the current day.
2. Real-time prediction for next hour demand based on historical data up to the current hour.

Further, we also consider:

1. Redefine training and testing data with a notion of a strict anchor time, i.e., no interleaving.
2. Reevaluate learning algorithms with data splitting by anchor time.
3. Improve the forecasting by considering the correlation structure of bike rental demand.
4. Test forecasting models under different business requirements.

2 Exploratory Data Analysis

2.1 Seoul Bike Sharing Demand Data

As mentioned earlier, we downloaded the data from the UCI Machine Learning Repo and it contained 14 measurements, split as 13 covariates and one supervisor, the rented bike count.

Variable	**Data Type**
DateTime	Date Time
RentedBikeCount	Numeric
Hour	Numeric
Temp	Numeric
Humidity	Numeric
Windspeed	Numeric
Visibility	Numeric
DewPointTemp	Numeric
SolarRadiation	Numeric
Rainfall	Numeric
Snowfall	Numeric
Seasons	Factor (4 Levels)
Holiday	Factor (2 Levels)
FunctionalDay	Factor (2 Levels)

2.2 Time Series Data

Fundamentally, our data is time-series data (Figure 1). As such, let y_t be the time series we're working to model, i.e., Seoul's bike sharing data.

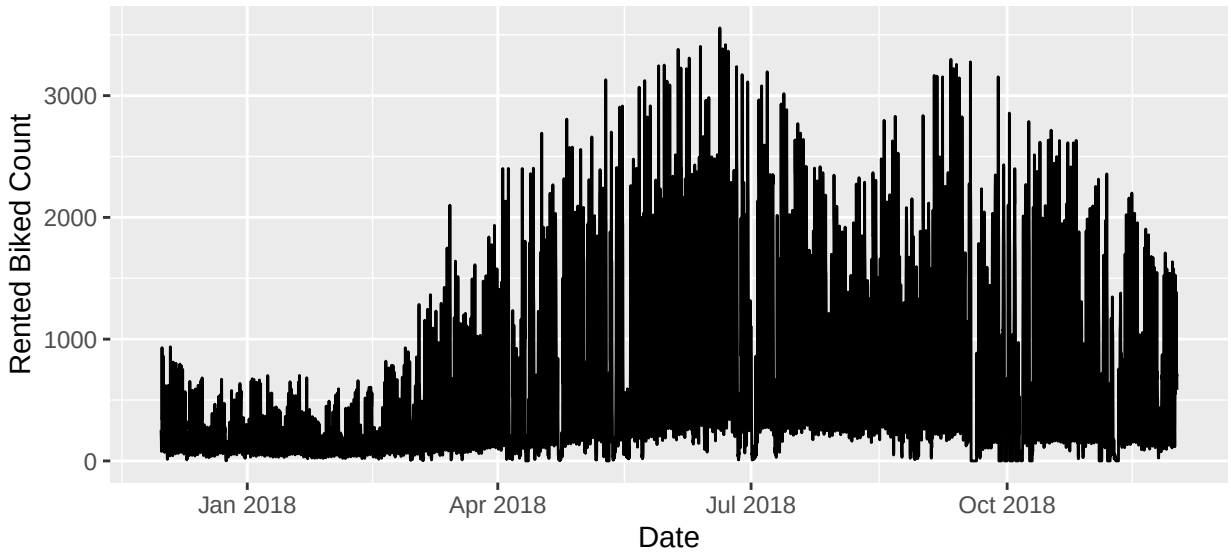

Figure 1: Hourly Rented Bike Count Over Entire Time Period

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 (Greene 2003). 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).

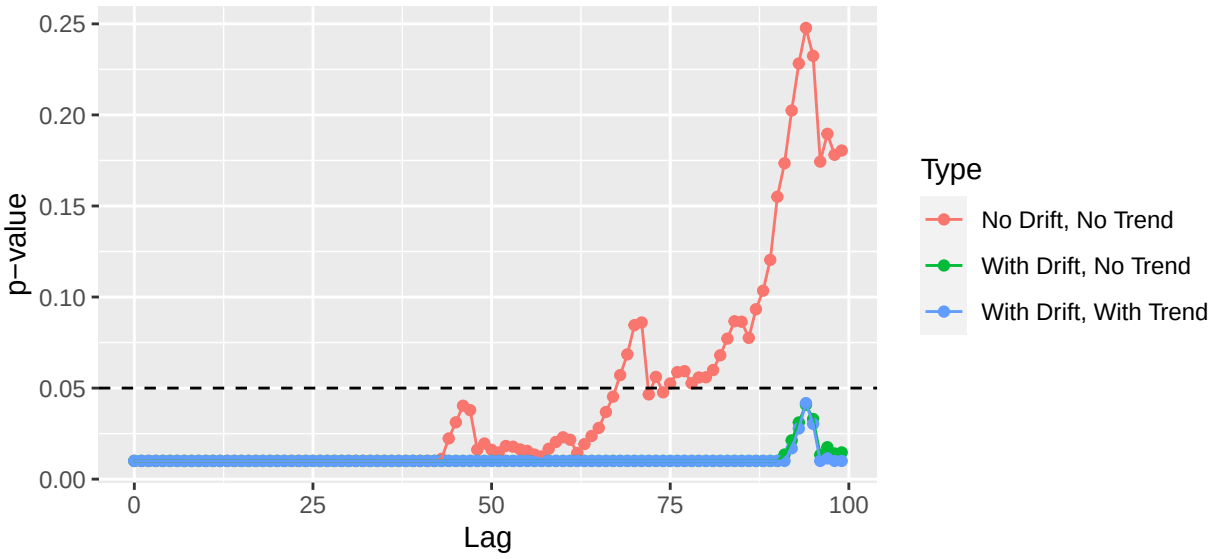

Figure 2: Augmented Dickey Fuller (ADF) Test for Stationarity

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}, while 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., $\tilde{y} = \{y_{t-1}, y_{t-2}, \dots, y_{t-k+1}\}$ (Box and Jenkins 1976). 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.

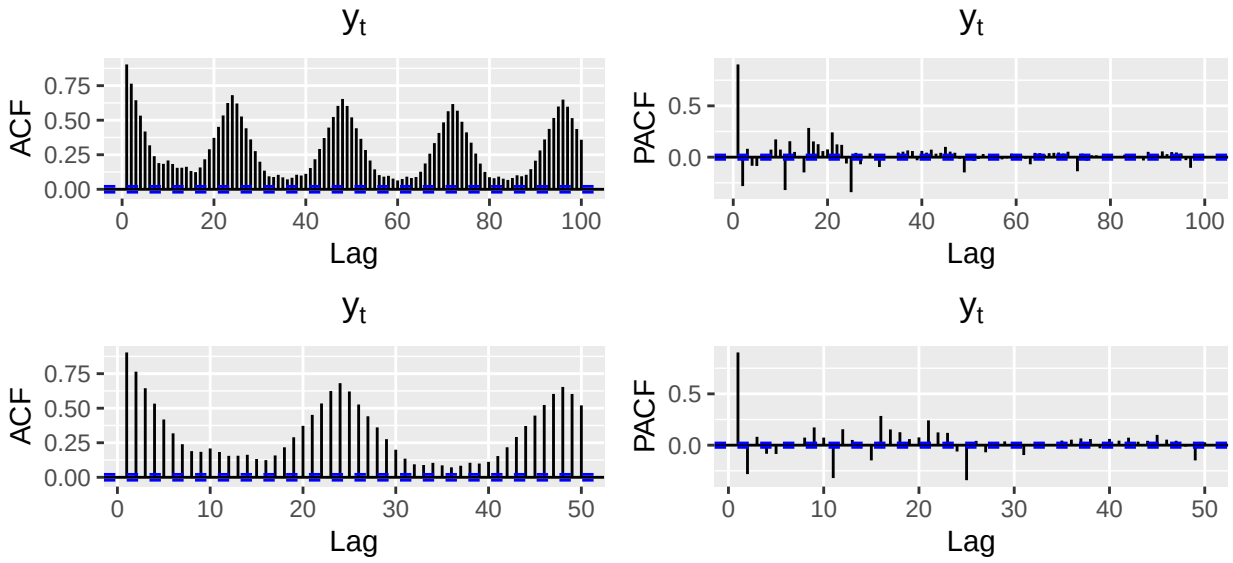

Figure 3: Autocorrelation and Partial Autocorrelation of Rented Bike Count, i.e., y_t

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 $h > q$ (Shumway and Stoffer 2019). 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

Figure 4 shows 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.

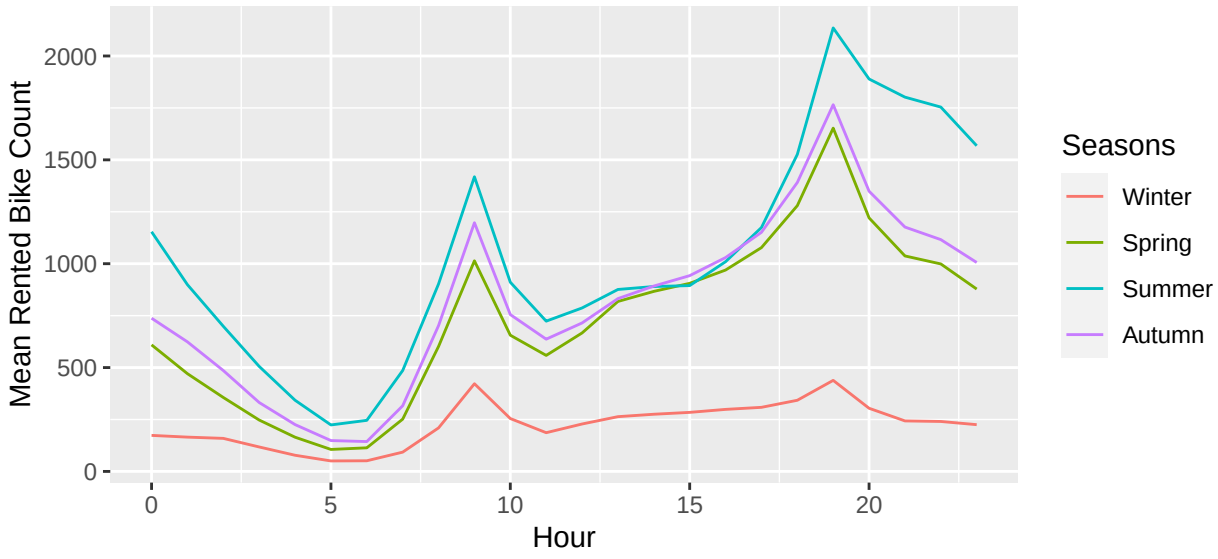

Figure 4: Rented Bike Count by Hour Grouped by Seasons

2.3.2 Qualitative Variables

Grouping by the qualitative variables, i.e., the factors, note:

- Figure 5 shows more rented bike count in non-holidays than holidays except for summer.
- If functional day is "no", there's not any bike rentals (Figure 6).

- Considering seasons and grouping by day of week, rented bike count by seasons isn't significantly affected by day of week (Figure 7).

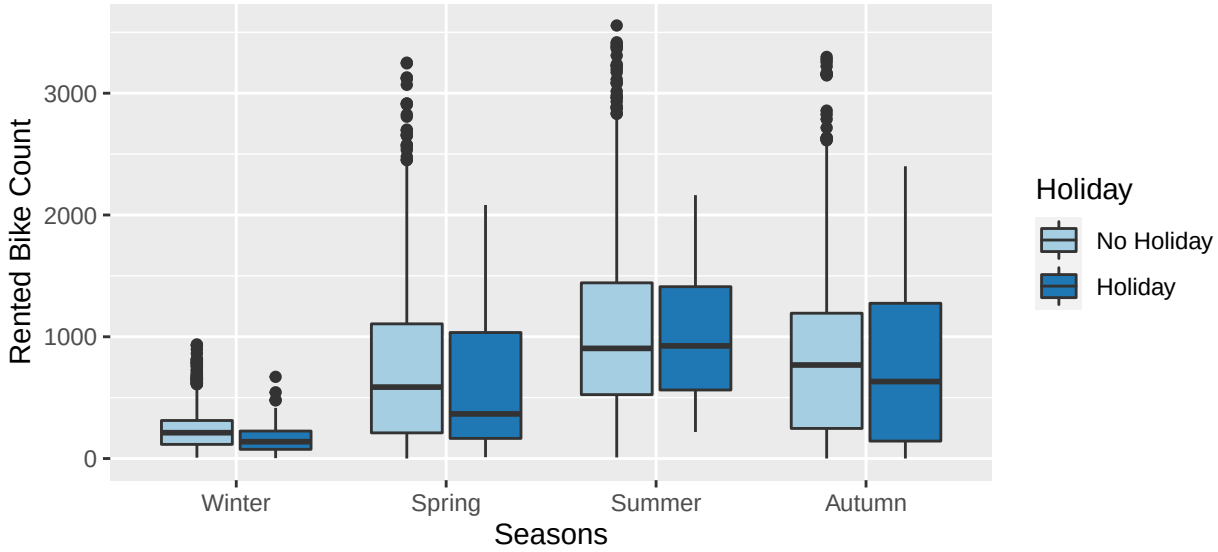

Figure 5: Rented Bike Count by Season Grouped by Holiday

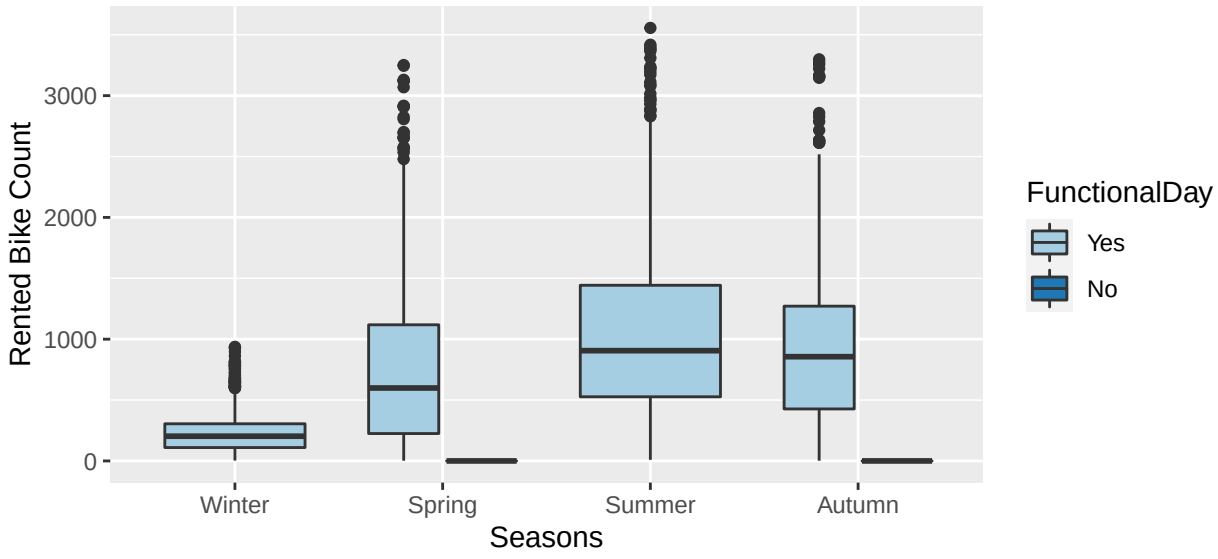

Figure 6: Rented Bike Count by Season Grouped by Functional Day

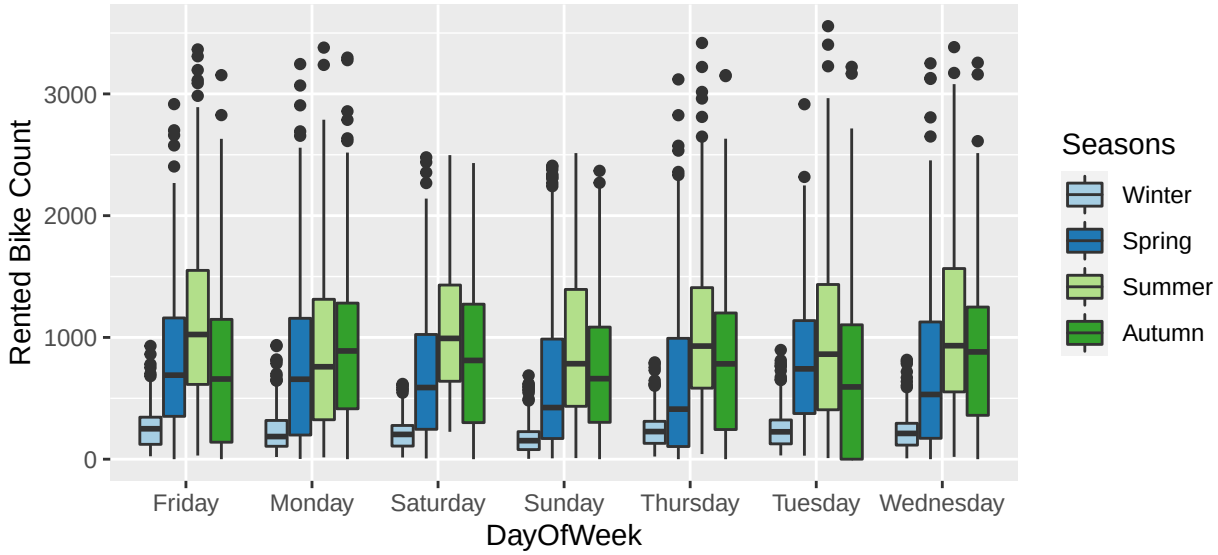

Figure 7: Rented Bike Count by Day of Week Grouped by Season

2.3.3 Quantitative Variables

Figure 8 and Figure 9 are showing correlations between quantitative variables and demand, for which we can note that:

- The covariance matrix shows `Temp`, `Hour` have relatively higher correlation with `RentedBikeCount` ($\rho_{x,y} > 0.4$).
- `DewPointTemp` and `SolarRadiation` have correlation greater than 0.2 ($\rho_{x,y} > 0.2$).
- `Temp` and `DewPointTemp` are highly correlated amongst each other $\rho_{x_1,x_2} > 0.9$).

- Since we don't see any covariate with $\rho_{x,y} > 0.5$, we state that there are no clear linear relationships between the supervisor and quantitative covariates.

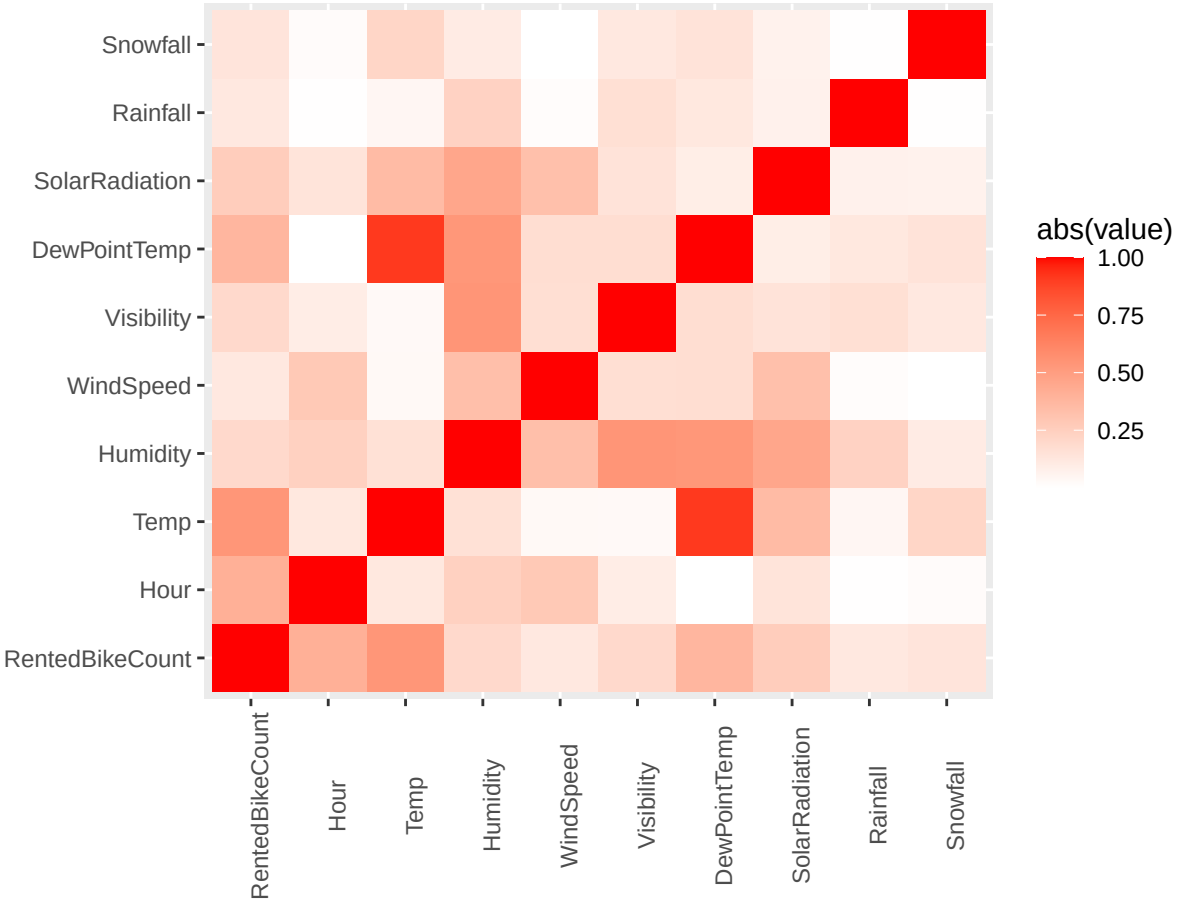

Figure 8: Covariates Correlation Matrix

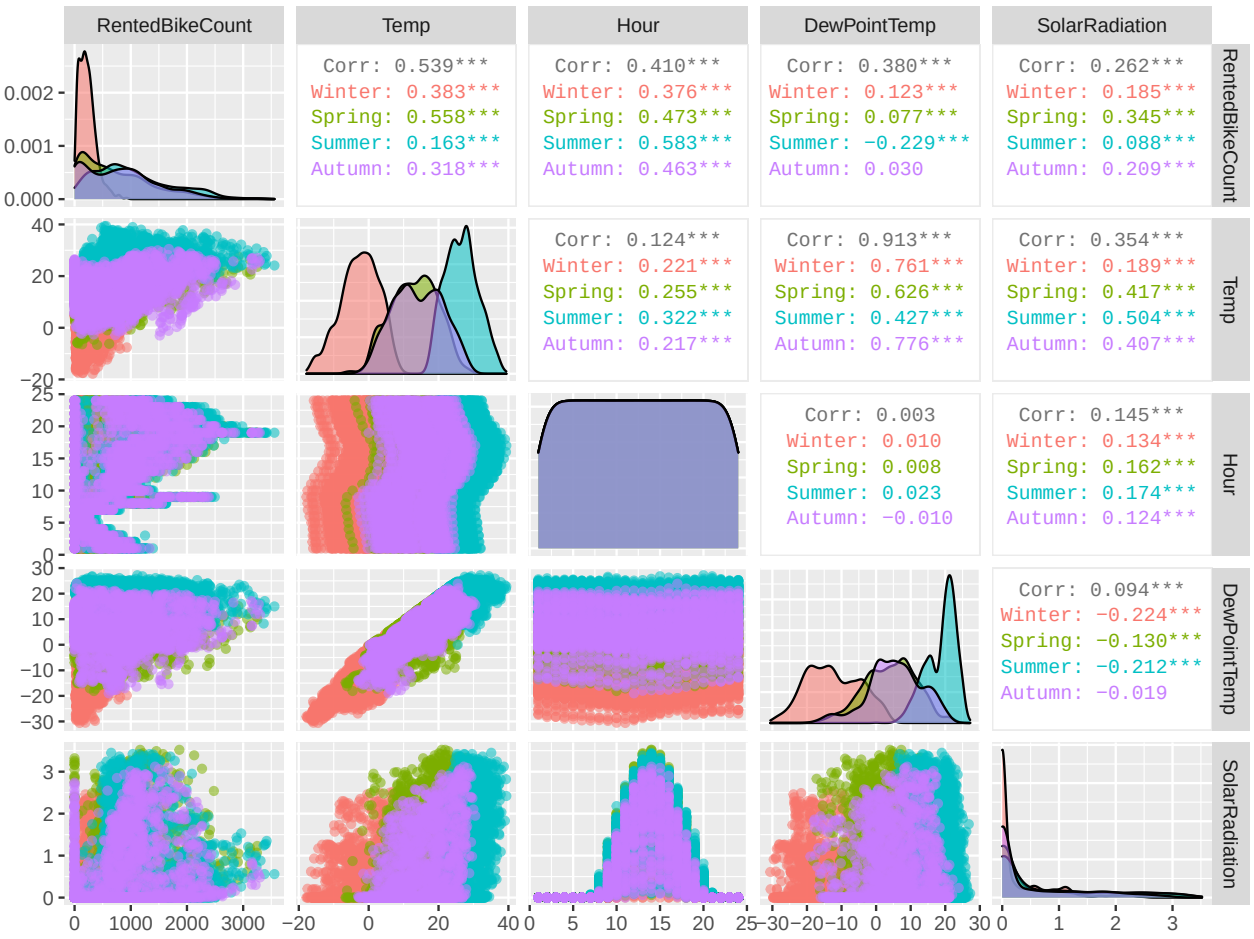

Figure 9: Top Four Covariates Correlation Matrix By Season

2.4 Splitting Training and Testing Data

The data set includes hourly bike rented count spanning a little over a year, from Dec 2017 to Nov 2018. 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 for which `FunctionalDay` = `No` before September 2018, which leaves little evidence for the model to identify the impact of `FunctionalDay` during training process if setting anchor date prior to September.

To maximize the training data size available while maintaining similar distributions, the testing anchor date and time will be set no earlier than November 1, 2018, where the training data feature distribution is close to all year feature distribution (See Figure 10-11 and Table 1-3).

2.4.1 Weather Information Distribution

Figure 10 and 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 qualitatively close enough distribution relative to the entire distribution.

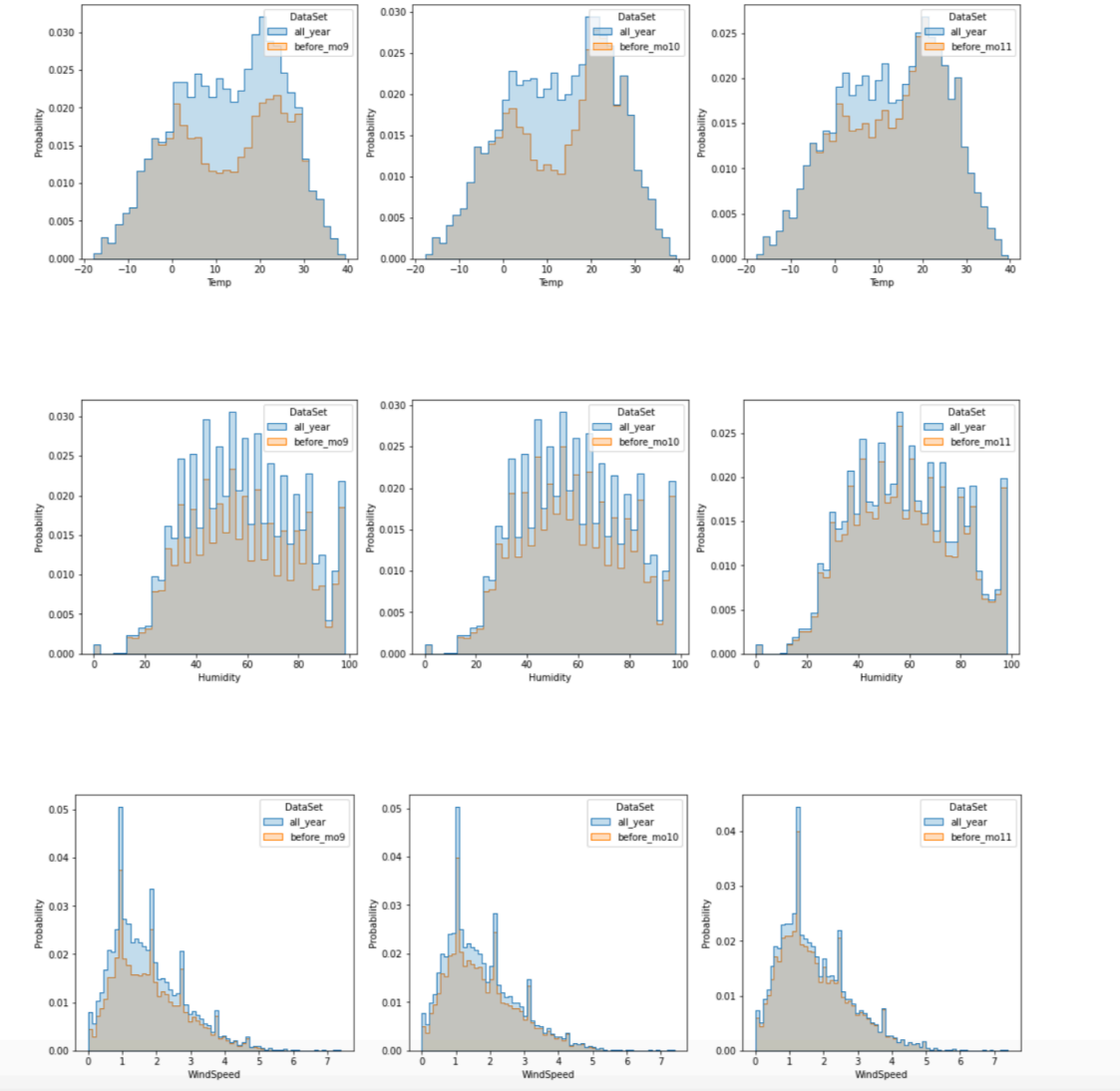

Figure 10: Temp, Humidity and WindSpeed Distribution

Table 1: Percentage of Observations by Season

	All Year	Before Sept.	Before Oct.	Before Nov.
Spring	0.25	0.34	0.3	0.27
Summer	0.25	0.34	0.3	0.27
Autumn	0.25	NaN	0.1	0.18
Winter	0.25	0.33	0.3	0.27

Table 2: Percentage of Observations by Holidays

	All Year	Before Sept.	Before Oct.	Before Nov.
No Holiday	0.95	0.95	0.95	0.95
Holiday	0.05	0.05	0.05	0.05

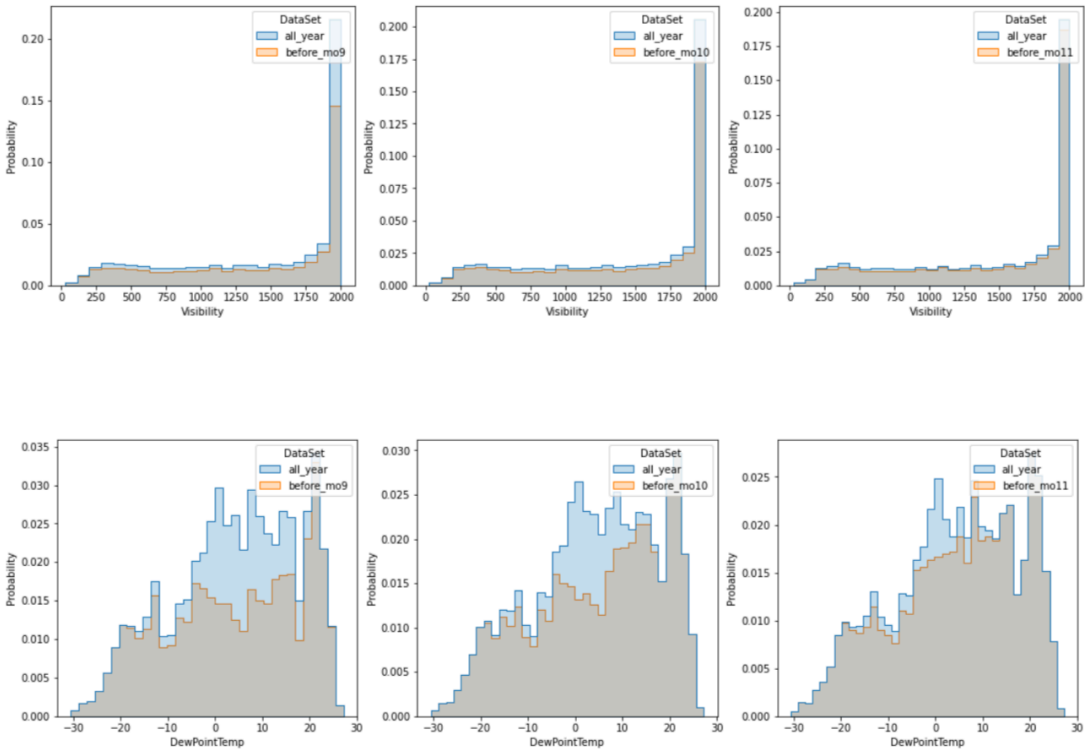

Figure 11: Visibility and DewPointTemp Distribution

2.4.2 Function Day and Holiday Distribution

Tables (1, 2 and 3) are showing number of observations by category in each data set. The last data set (setting anchor day at November 1, 2018) has quantitatively the closest proportion of `FunctionalDay` relative to the entire, non-subsetted data..

Table 3: Percentage of Observations by Function Days

	All Year	Before Sept.	Before Oct.	Before Nov.
Function Day	0.97	0.99	0.98	0.97
Non-Function Day	0.03	0.01	0.02	0.03

3 Learning Algorithms

On a high level, we wish to do supervised learning, i.e., learn the relationship, i.e., fit a model, between the supervisor y (`RentedBikeCount` here) and its covariates such that we can make predictions $\hat{y}$ that perform well beyond our period of learning (Russell and Norvig 2009). There are numerous learning algorithms we can consider and consequently we consider a subset containing both linear approaches and non-linear approaches.

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 (Berger 1985).

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. $\hat{f}$

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

3.1.4 Summary

In short, our goal is to consider different learning algorithms, "learn" the relationship between the data and the supervisor, and make the best predictions we can, as defined by jointly by our risk estimate and the embedded loss metric, squared error loss $\ell(\hat{y}, y) = (\hat{y} - y)^2$ in our case. We will estimate risk in two different ways, cross-validation to help guide our hyperparameter selection, and "test error" as the final hold-out to evaluate the "tuned" hyperparameters (Stone 1974).

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, $\hat{\beta} = \arg\min_\beta ||y - X\beta||_2^2$ (Sheather 2009).

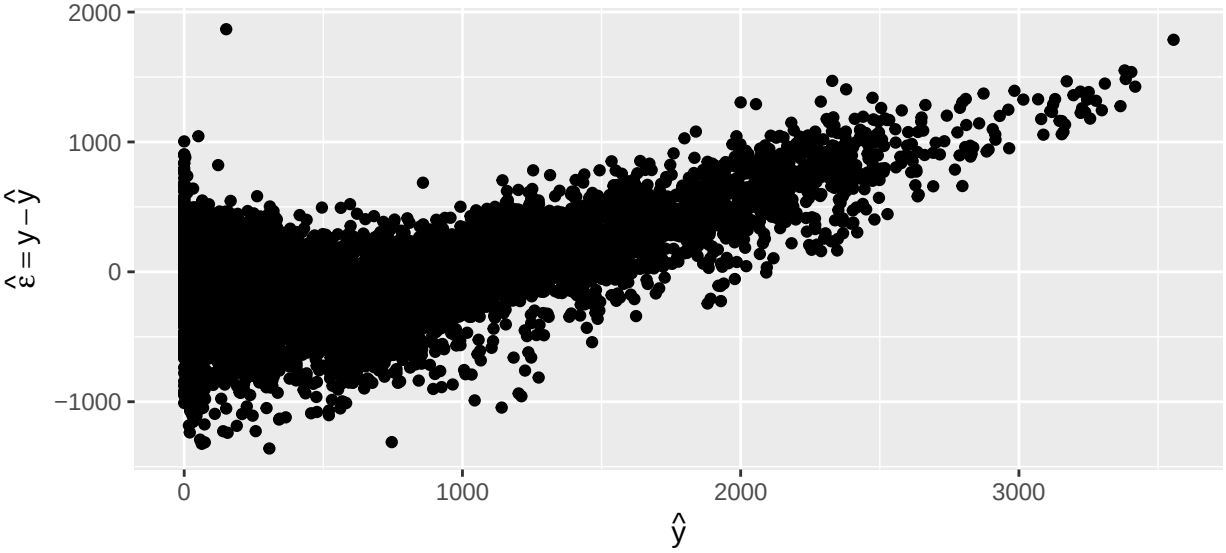

Figure 12: Residual Plot for Linear Regression

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

From Zou and Hastie (2005), Elastic Net is an extension of Linear Regression, where we do a mixture 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, another parameterization is having a singular λ and a "mixing ratio" between L_1 and L_2 regularization in the form of α. Then, $\hat{\beta}(\lambda, \alpha) = \arg\min_\beta \left(||y - X\beta||_2^2 + \lambda((1-\alpha)||\beta||_2^2 + \alpha||\beta||_1)\right)$. 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 choose hyperparameters with the lowest k-fold cross-validation risk estimates (Efron 2004).

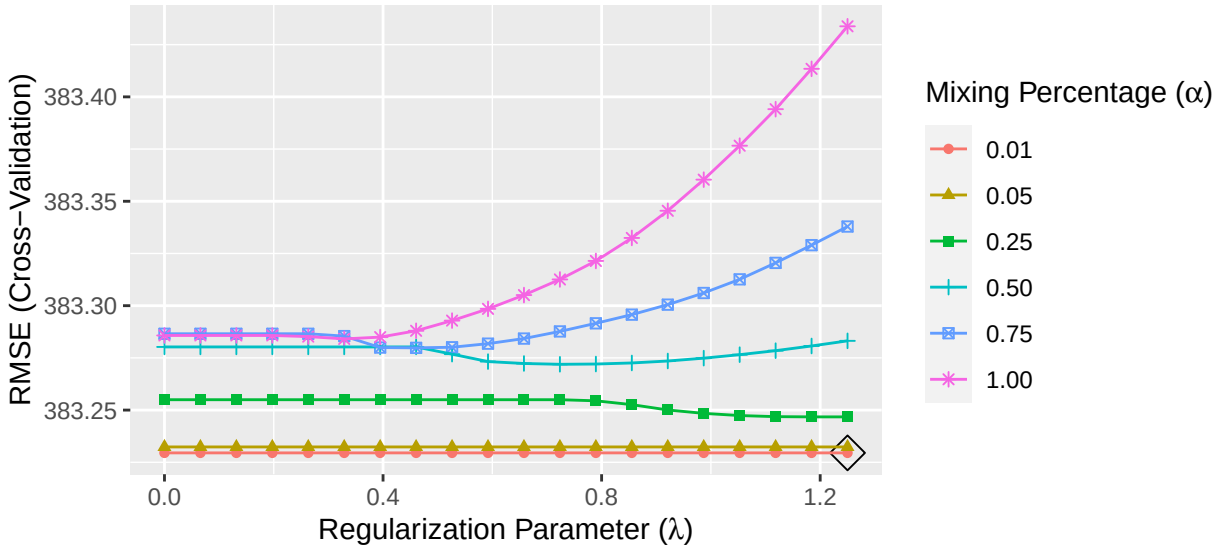

Figure 13: Cross-Validation Risk Estimates for Elastic Net Hyperparameters

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.

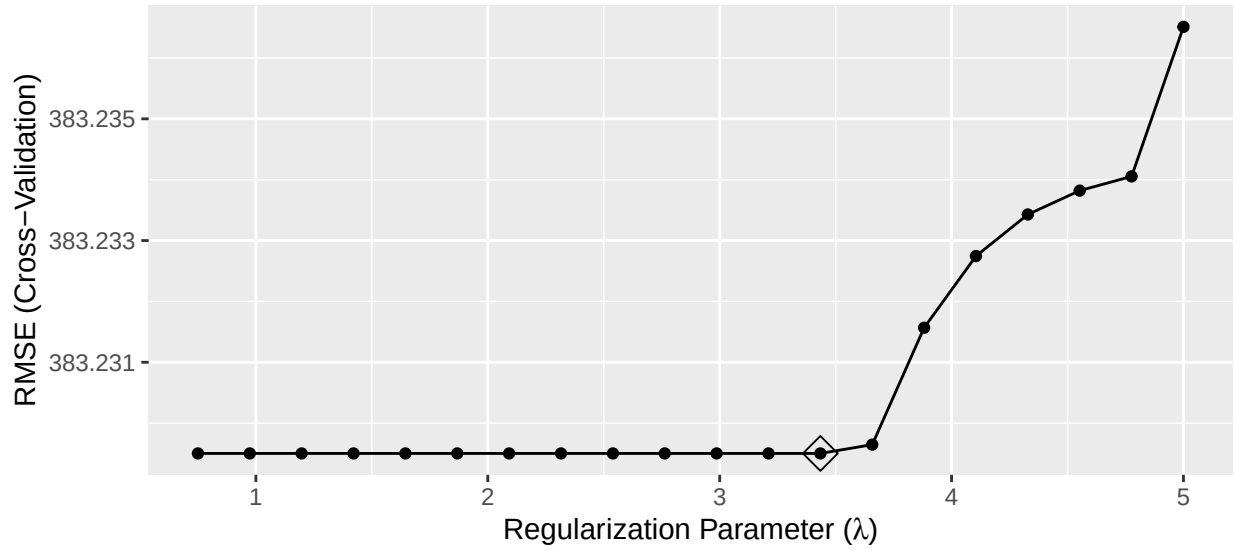

Figure 14: Cross-Validation Risk Estimates for Elastic Net Hyperparameters Past Boundary Condition

Thus, the hyperparameters that minimize the cross-validation risk estimate is $\hat{\alpha} = 0.01$ and $\hat{\lambda} = 3.43$.

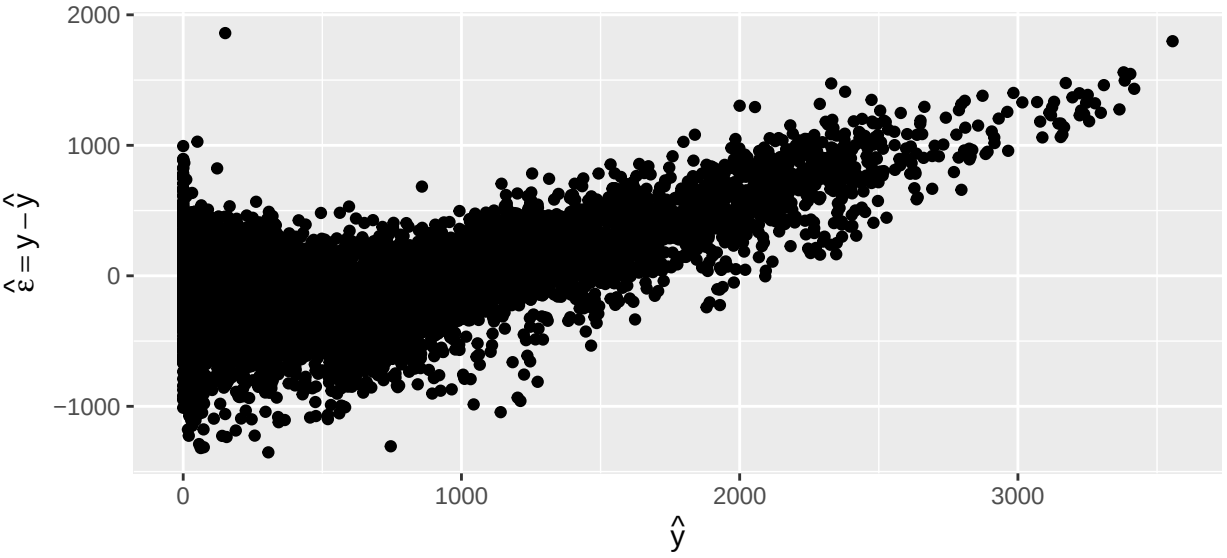

Figure 15: Residual Plot for Elastic Net

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)

Per Friedman (1991), 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 covariates 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 to consider as well as the degree of interaction are the hyperparameters for MARS.

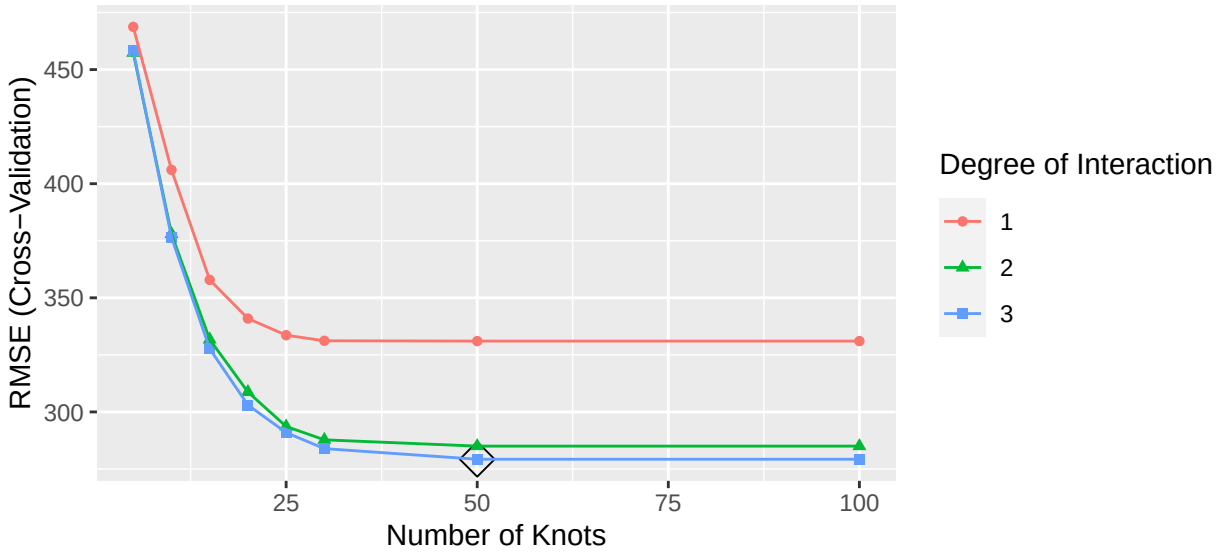

Figure 16: Cross-Validation Risk Estimates for MARS Hyperparameters

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 there seem to be diminishing return as there's not a large difference between allowing 3-way interactions relative to 2-way interactions.

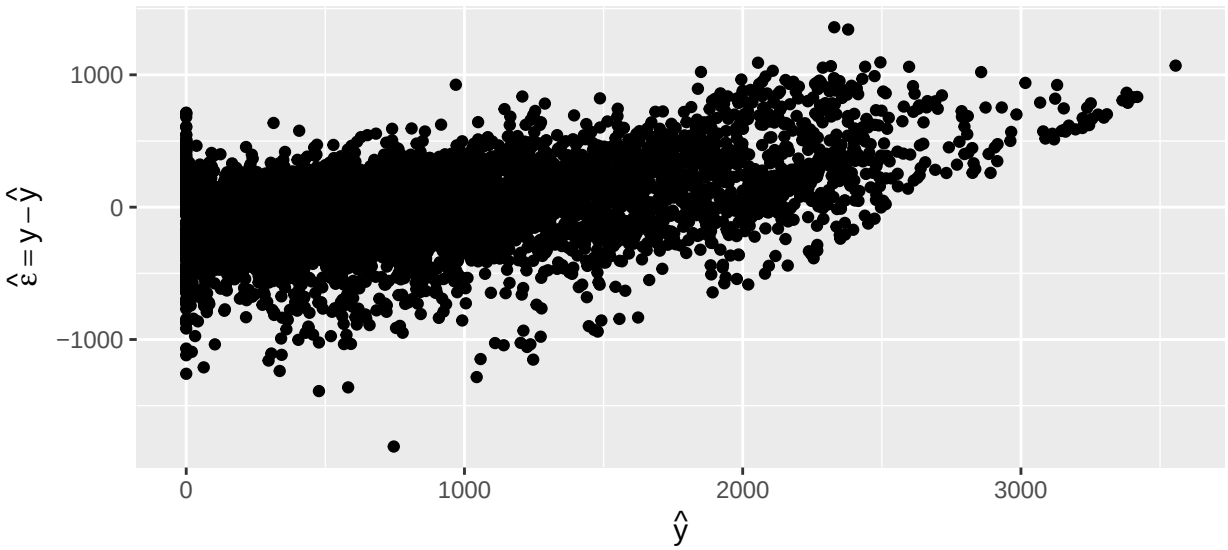

Figure 17: Residual Plot for MARS

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

3.3.2 Decision Tree

Per Loh (2011), Decision Tree is an umbrella term for machine learning methods that recursively partitions the data and fits a simple prediction model at each partition, for which the results can be viewed as a decision tree. We specifically use a variant of CART (L. Breiman et al. 1984), which imposes a `cp`, complexity parameter, effectively specifying a minimal amount of improvement needed for each split, thus saving later tree pruning that is part of CART's regular routine (Therneau and Atkinson 2019).

As such, the only hyperparmeter we have to consider is `cp`.

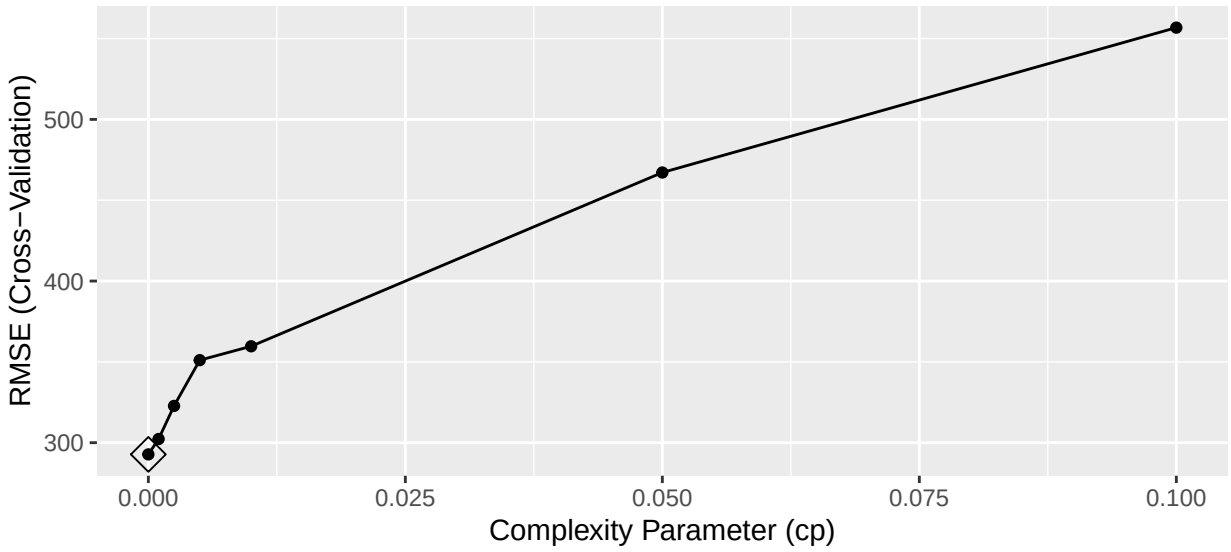

Figure 18: Cross-Validation Risk Estimates for Decision Tree Hyperparameters

Here, we can see that as we increase the hyperparameter `cp`, the cross-validation risk-estimate is increasing. This suggests that our best tree is a fully growned decision tree and that any pre-pruning with `cp` would be detrimental.

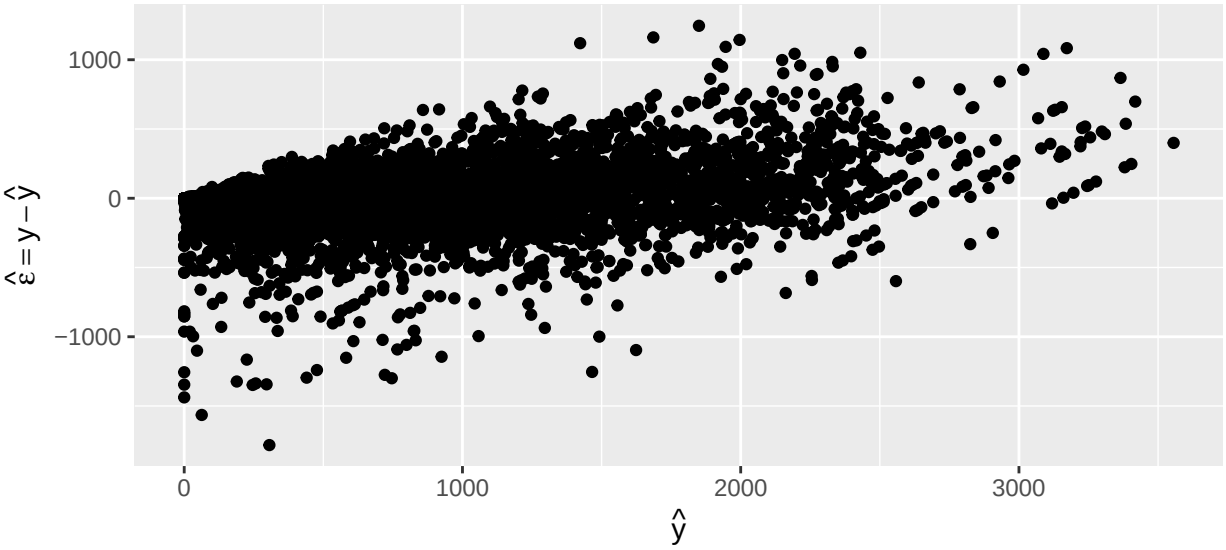

Figure 19: Residual Plot for Decision Tree

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

Random Forest is an ensemble of trees, where for each tree we use a bootstrap of our original data and further for each decision node for that tree, we only consider a subset of covariates, e.g., $\sqrt{p}$. The motivation for Random Forest is that Decision Trees are low bias high variance procedures and by doing the aforementioned approach, we're able to maintain the bias but lower the variance (Leo Breiman and Schapire 2001).

We fix the number of trees in the ensemble to be 500, and the hyperparameters we consider are `mtry`, which specifies the number of covariates to use, and `min.node.size`, which specifies the minimum amount of observations in a terminal node.

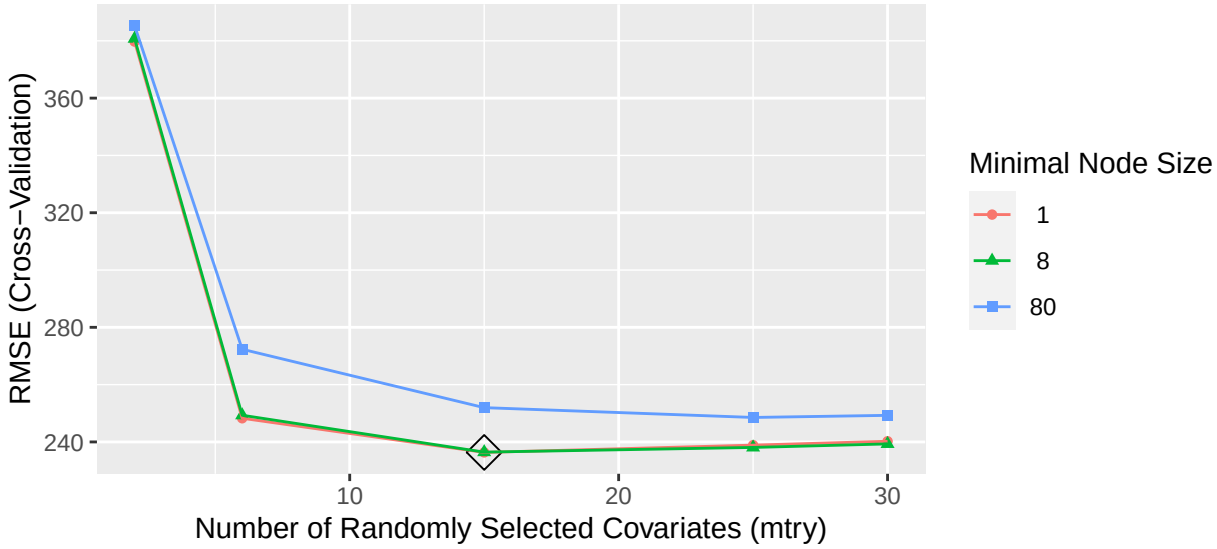

Figure 20: Cross-Validation Risk Estimates for Random Forest Hyperparameters

Thus, the hyperparameters that minimize the cross-validation risk estimate is `mtry` $= 15$ and `min.node.size` $= 1$.

3.3.4 Boosting

From Friedman, Hastie, and Tibshirani (1998), Boosting is a forward stagewise additive model in a set of elementary "basis" functions, e.g., weak classifiers $G_m(x) \in \{-1, 1\}$. Mathematically, we can frame this as $f(x) = \sum_{b=1}^{B} \beta_b b(x; \gamma_b)$, where β's are the expansion coefficients, and $b(x; \gamma) \in \mathbb{R}$ are simple functions of multivariate argument x and characterized by γ (Hastie, Tibshirani, and Friedman 2001).

XGBoost is a popular implementation of Boosting that also extends it to be computationally faster as well as more scaleable, across multiple cores as well as multiple computers (Chen and Guestrin

2016).

Assuming we use a decision tree as our "basis" function, hyperparameters we consider are how many rounds of boosting (`nrounds`), the maximum depth of the tree (`max_depth`), and what fraction of covariates to consider per tree (`colsample_bytree`), similar to Random Forest.

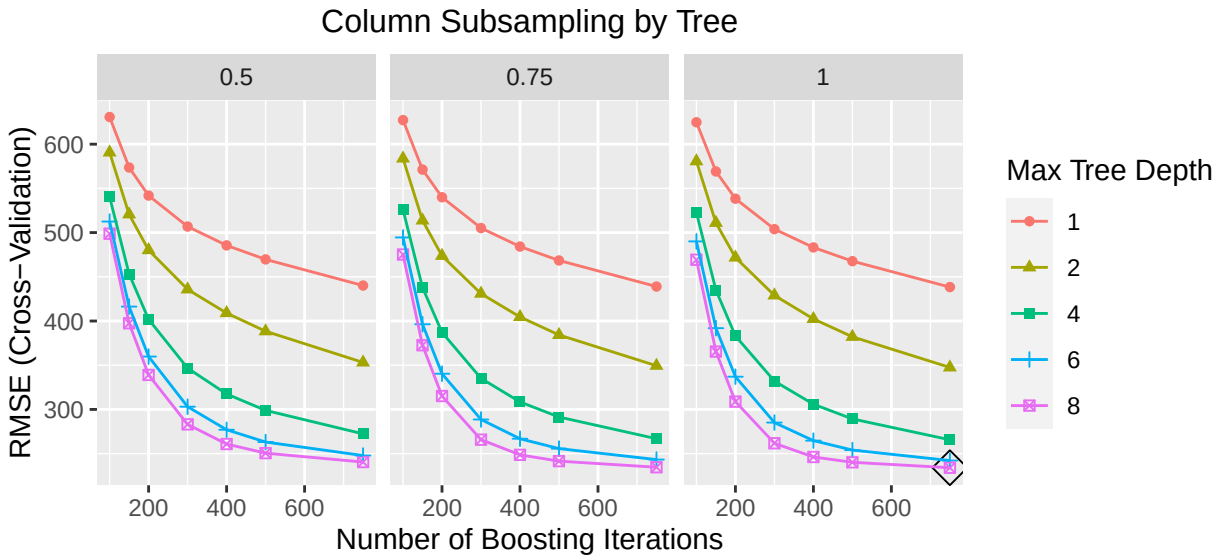

Figure 21: Cross-Validation Risk Estimates for Boosting Hyperparameters

Thus, the hyperparameters that minimize the cross-validation risk estimate is `nrounds` $= 750$, `max_depth` $= 8$, and `colsample_bytree` $= 1$. That being said, the big driver is `max_depth`, i.e., the inverse relationship between cross-validated risk estimate and `max_depth` suggests having increasingly non-linear interactions are helpful.

3.4 Evaluation

Thus far, we've fit the different learning algorithms using our in-sample data and have used cross-validation risk estimates to determine the appropriate hyperparameters. But, to select amongst the learning algorithms, we need to use data that we haven't seen thus far, i.e., the test data. From this, we can form a new risk estimate $\hat{R} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2$, i.e., test error.

But, since test error isn't bounded and can thus be hard to conceptualize, we also use an out-of-sample R^2,

$$R^2_{OOS} = 1 - \frac{\sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n_{\text{test}}} (y_i - \bar{y}_i)^2}$$

Table 4: Learning Algorithm Comparison

	R^2_{OOS}	**Test Error**
Linear Regression	0.58	97959.25
Elastic Net	0.59	95206.58
MARS	0.67	76580.03
Decision Tree	0.40	139592.56
Random Forest	0.73	63833.75
Boosting	0.76	56350.83

Note that R^2_{OOS}, unlike R^2 which is traditionally used on train data and is bounded between 0 and 1, and can be negative. Conceptually $R^2_{OOS} < 0$ implies that using the out-of-sample average for y, i.e., $\bar{y}_{OOS}$, would have been more performant.

From Table 4, we can see that the Boosting learning algorithm has the highest R^2_{OOS} at 0.76. As such, when going forward and considering additional covariates and more complex models, we will fix our learning algorithm to use Boosting.

4 Correlation Structure

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. (See Figure 22). However, this past demand information is not well incorporated in the set of covariates used in our statistical models. A simple way to use the information is to add dependency features of past demand.

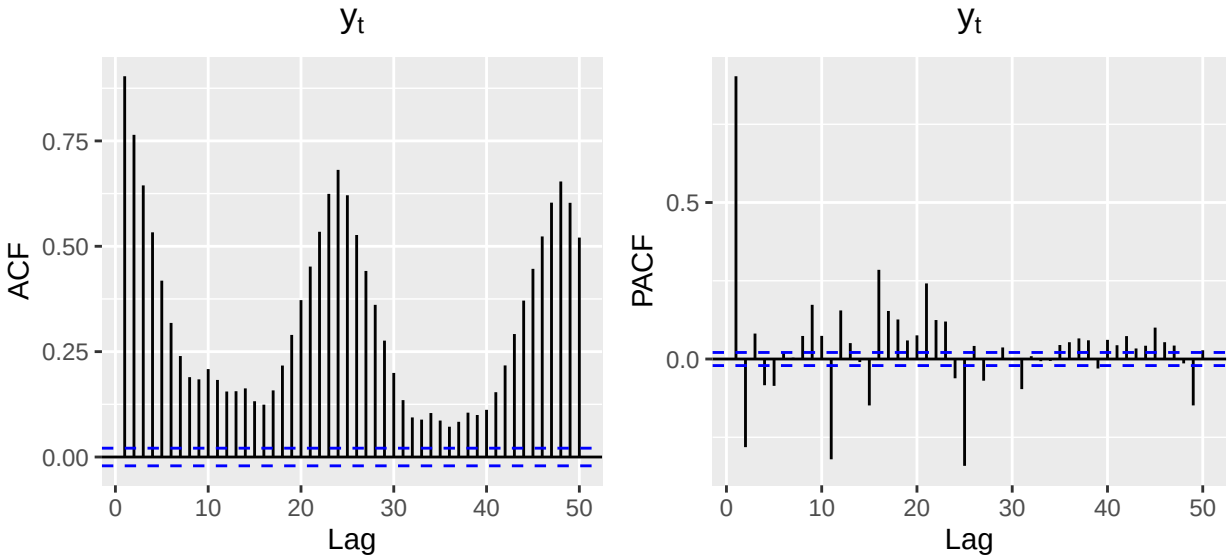

Figure 22: Autocorrelation and Partial Autocorrelation of Rented Bike Count, i.e., y_t

4.2 Dependency Feature

In addition to weather feature and hour feature, dependency features for past demand are added, e.g., `lag_1` means demand in the last hour; `lag_4` means hourly demand 4 hours ago, etc. Note, by incorporating lagged values of our covariates, some of our observations will have missing measurements, e.g., the first observation. As such, any observations with induced missing measurements will be deleted from training data.

4.2.1 Estimation without Dependency Features

In section 3.4, the Boosting learning algorithm performed best with respect to R^2_{OOS} relative to the other learning algorithms considered. Therefore, for all future model training and predictions, we will use the Boosting learning algorithm along with the tuned hyperparmeters we identified in

section 3.3.4. All observations before anchor date Nov 1, 2018, exclusive, will be used for training, and the 30 observations from Nov 1, 2018 to Nov 30, 2018, inclusive, are used for testing. In order to compare the impact of dependency features, we fit a baseline model M_0 that only has weather features and hour information, without any dependency features. The R^2_{OOS} in M_0 is 0.756. Figure 23 is showing the feature importance of M_0, which suggests `Temp` as the most important features affecting the demand.

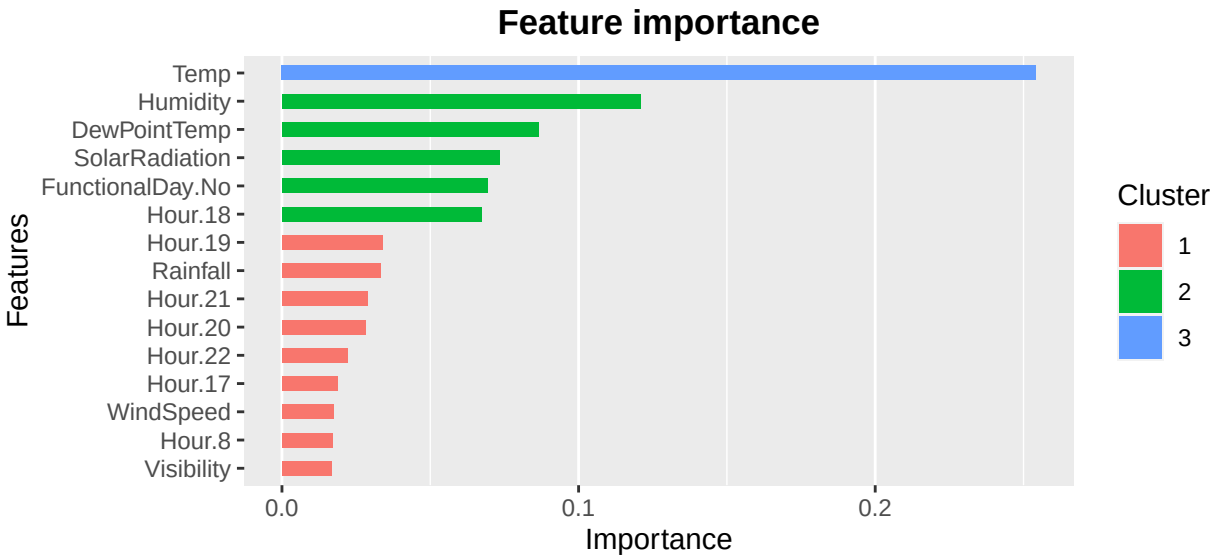

Figure 23: Feature Importance M_0

4.2.2 Estimation with Dependency Features

Based on PACF plot (Figure 22), dependency features within one 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 R^2_{OOS} goes up to 0.967 - a significant improvement from M_0. In addition, the feature importance plot (Figure 24) for M_1 below shows the past demand features are more important than weather features compared with M_0.

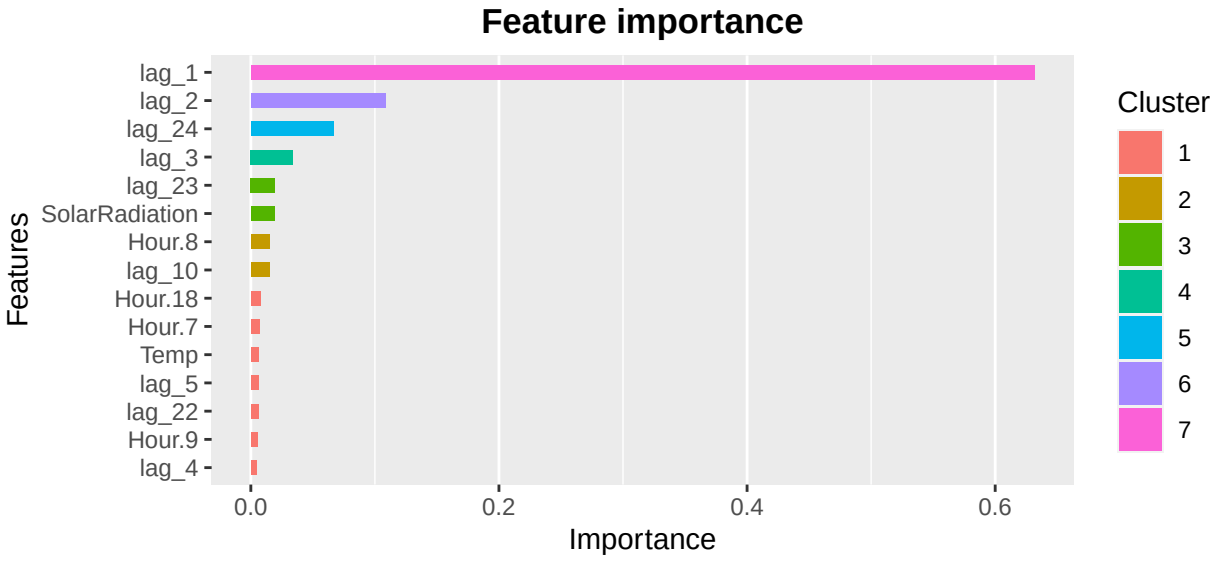

Figure 24: Feature Importance M_1

4.2.3 Estimation with Reduced Dependency Features

Based on the feature importance plot (Figure 24) of M_1 and the PACF plot (Figure 22), the dependency features can be reduced to lag 1, 2, 11, 16 and 24 to reduce the number of covariates considered, i.e., a more parsimonious model. Thus M_2 is fitted with smaller number of dependency covariates and the R^2_{OOS} is 0.958, which is less than one percentage point 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 considered are 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 and 2 days ago and 1 week ago. The R^2_{OOS} is 0.95, which is as good as M_1 and M_2.

4.2.5 Estimation Comparison

Table 5 is comparing the R^2_{OOS} among models with different dependency features. Considering the test accuracy and variable size, M_2 and M_3 are preferred.

Table 5: Test R^2 Comparison of Models with Different Dependency Features

	Model	R^2_{OOS}
No Dependency Feature	M_0	0.756
Full Dependency Feature	M_1	0.967
Reduced Dependency Feature	M_2	0.958
Business Dependency Feature	M_3	0.950

5 Forecasting Application

5.1 Business Scenaros

A purpose of predicting bike rental 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 restricted to be updated on a daily basis at midnight, and we're required to analyze and forecast the next 24 hours' **hourly demand** in order to do high-level planning for 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 hours, hour by hour.

5.1.2 Real-time Data Update

If the system infrastructure could support real-time data updates, an hourly model training could be run to predict the next **hour's demand**. Any changes in the past hour could be incorporated in the statistical model and implicitly used in the next hour demand prediction. To test the performance, new models will be trained hourly with all the data available at that moment (including most recent demand data from last hour) and used to predict the demand in the next coming hour.

5.2 Hourly Demand Forecasting with Daily Data Update

5.2.1 Estimator and Dependency Features

Conceptually, when data is updated once every day, the latest observation available to use as dependency feature is the lag 24 of rented bike count at the interested prediction time steps. Therefore, the smallest lag of dependency features can be added is lag 24. Based on the business assumption

Table 6: Forecasting Result with Daily Data Update and Dependency

Train R^2	**Train CV** R^2	**Test** R^2
0.977	0.799	0.86

in dependency feature study, lag 24, 48 and 24 $\times$ 7 are added as dependency features, which represent the past demand from the same hour 1 day ago, 2 days ago and 1 week ago. The model training is repeated daily for November 2018. And the learning algorithm used is Boosting with the hyperparameters identified in section 3.3.4.

5.2.2 Forecasting Results

When considering a daily data update from the business, there are 30 statistical models we fit and use to predict for the next day, one for 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.

Table 6 shows the average training R^2 over the 30 iterations is 0.97, the average mean CV R^2 over the 30 iterations is 0.79, and the test R^2 over the 30 iterations is 0.86. There's no large discrepancy between the cross validation R^2 and test R^2. Therefore, there's no strong evidence of overfitting.

With daily data update, some poor predictions exist in certain iterations, i.e., days. This is seen clearly in Figure 25 when comparing forecast demand and real demand. On Nov 3, Nov 6 and Nov 9, when there's no demand due to it being a non-functional day, the forecasting at the beginning of the day still predicts a certain amount of demand, since at prediction time, we don't know if it will be a functional day or not.

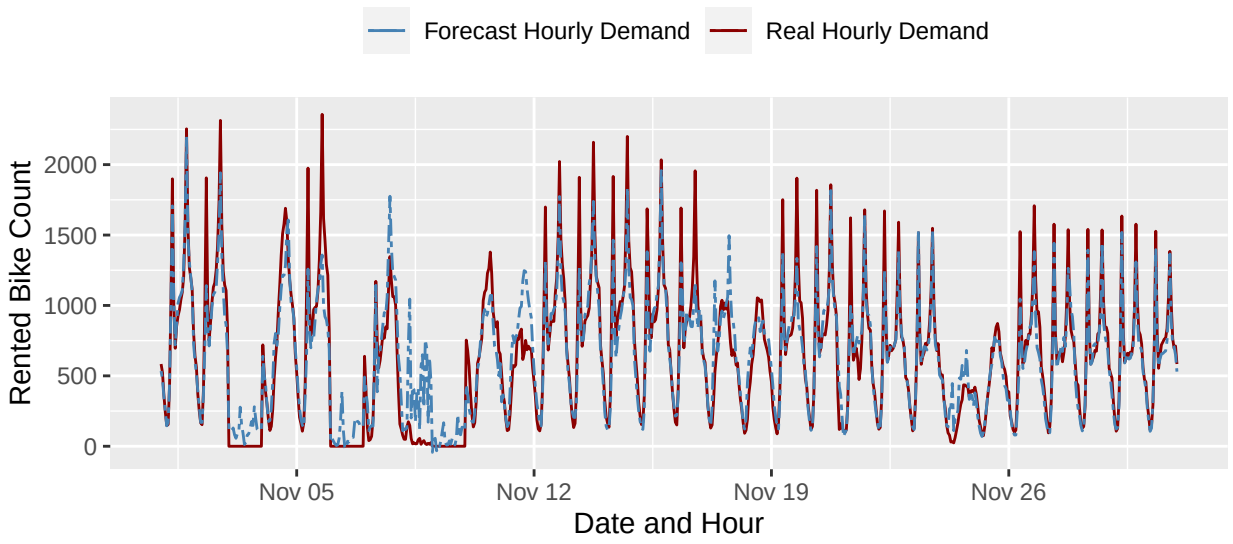

Figure 25: Forecasted Demand and Real Demand Comparison with Daily Data Update and Dependency

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, 2 days ago and 1 week ago as dependency features to the data (M_3 in section 4.2.4). The model fitting is repeated every hour and used to predict demand only for the coming hour. And the learning algorithm used is Boosting with the hyperparameters identified in section 3.3.4, as before.

5.3.2 Forecasting Results

In the real-time data update, there are 30 $\times$ 24 repeated model fitting and forecasting (for each day and hour). In each iteration of model training and forecasting, all observations prior to the iteratorion's date and hour are used as training data and the next 1 hour demand is used as testing data.

Table 7 shows both train R^2 and average mean CV R^2 over the 30 $\times$ 24 iterations are above 0.9. The test R^2 reaches 0.96. And the figure comparing forecast demand and real demand (See Figure 26) 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 latest demand in the past hour and adjust the coming hour demand prediction, e.g, forecast demand in Nov 3, Nov 6 and Nov 9 stay low when detecting low demand in the previous hours.

Table 7: Forecasting Result with Real-time Data Update and Dependency

Train R^2	**Train CV R^2**	**Test R^2**
0.995	0.929	0.964

Table 8: Forecasting Results Comparison with Dependency

	Daily Data Update	**Real-time Data Update**
Train R^2	0.977	0.995
Train CV R^2	0.799	0.929
Test R^2	0.860	0.964

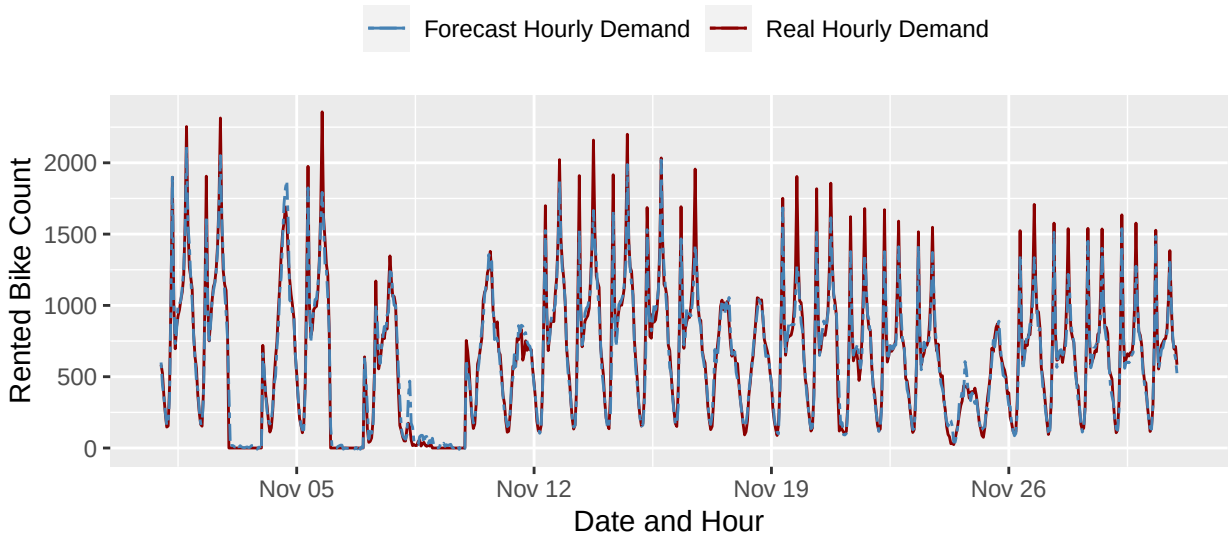

Figure 26: Forecasted Demand and Real Demand Comparison with Real-time Data Update and Dependency

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 percentage point improvement (see Table 8). Comparing the forecast demand to real demand, the real-time data update forecasting shows much lower discrepancies than the daily data update forecasting (See Figure 27 and 28).

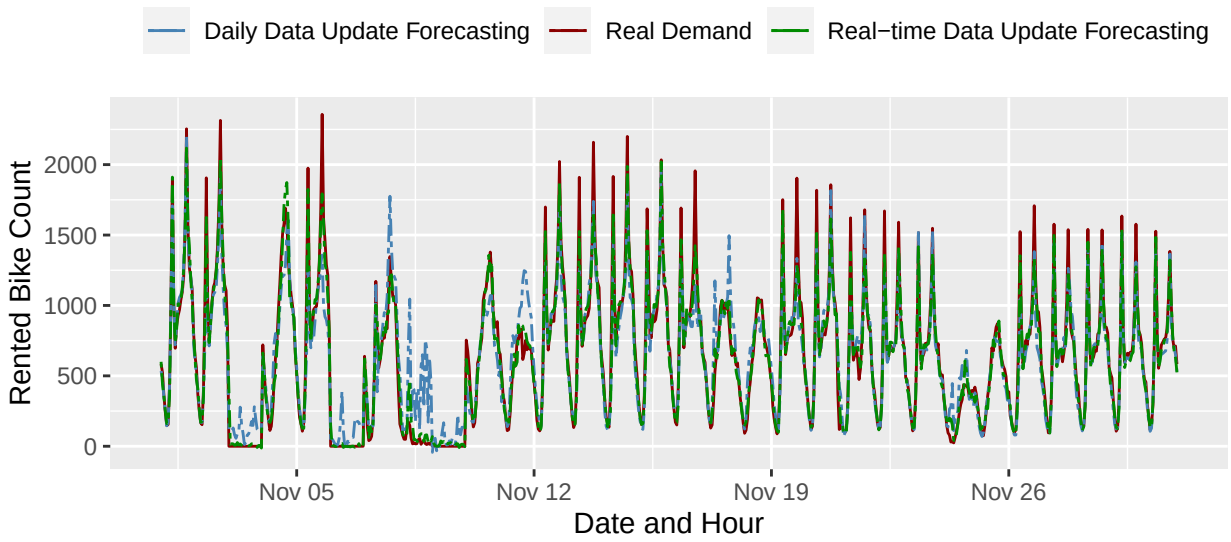

Figure 27: Forecasted Demand Comparison with Dependency

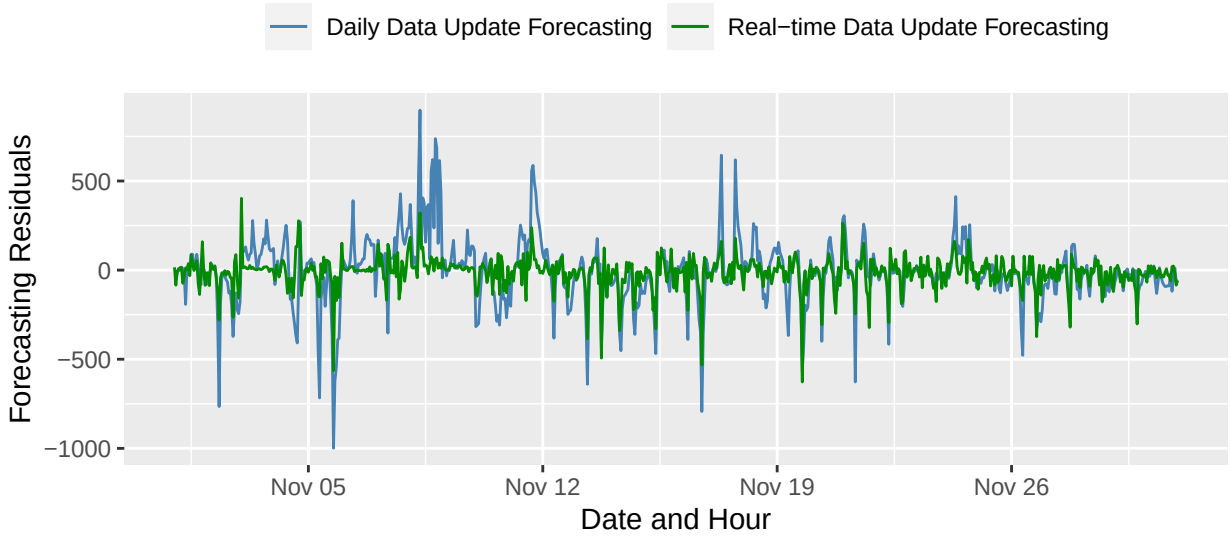

Figure 28: Forecasting Residual Comparison with Dependency

5.4.2 Improvement with Correlation Structure

To better understand the importance of dependency induced by the correlation structure, the same forecasting studies (repeated model training and forecasting on daily basis and hourly basis) are conducted without the dependency features, similar to the M_0 model defined in section 4.2.1.

Comparing the statistical models with dependency covariates and the statistical models without dependency covariates, Table 9 shows 8.5 percentage point improvement in daily data update and 14.7 percent point improvement in real-time data update in terms of test R^2. Comparing the two

Table 9: Forecasting Results Comparison with and without Dependency

	Train R^2	**Train CV** R^2	**Test** R^2
Daily Update with Dependency	0.977	0.799	0.860
Real-time Update with Dependency	0.995	0.929	0.964
Daily Update No Dependency	0.960	0.686	0.775
Real-time Update No Dependency	0.960	0.686	0.817

figures plotting forecasting residuals (See Figure 29 and 30), there are more significant improvements by adding dependency features with real-time data update than the scenario of daily data update, i.e., the blue residual line in the second figure is qualitatively smoother 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 percentage point higher) forecasting accuracy than a real-time data update without dependency.

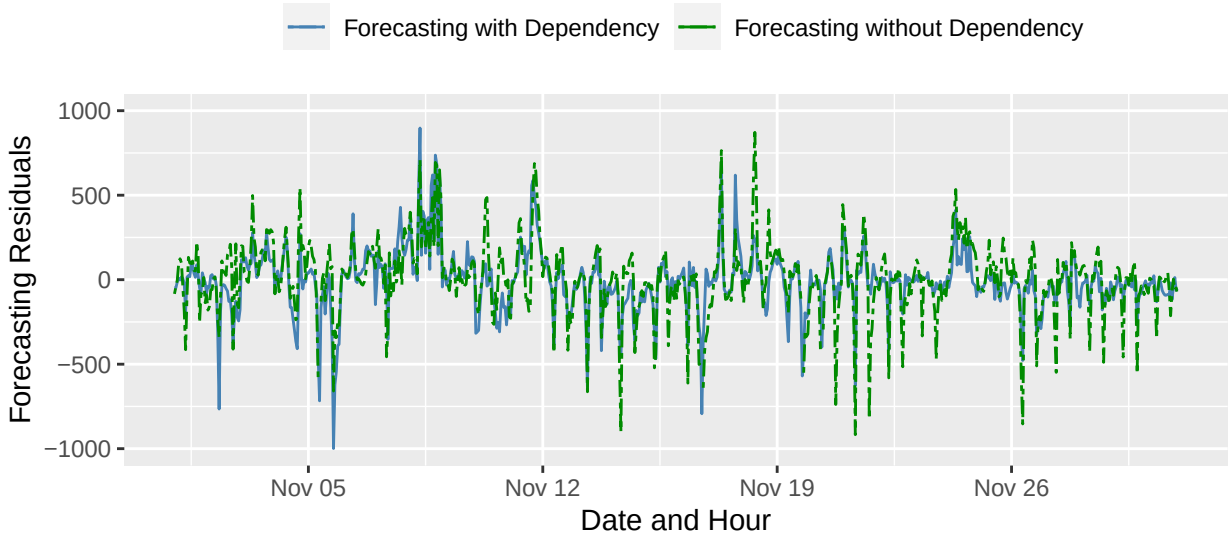

Figure 29: Forecasted Demand Comparison with Daily Data Update

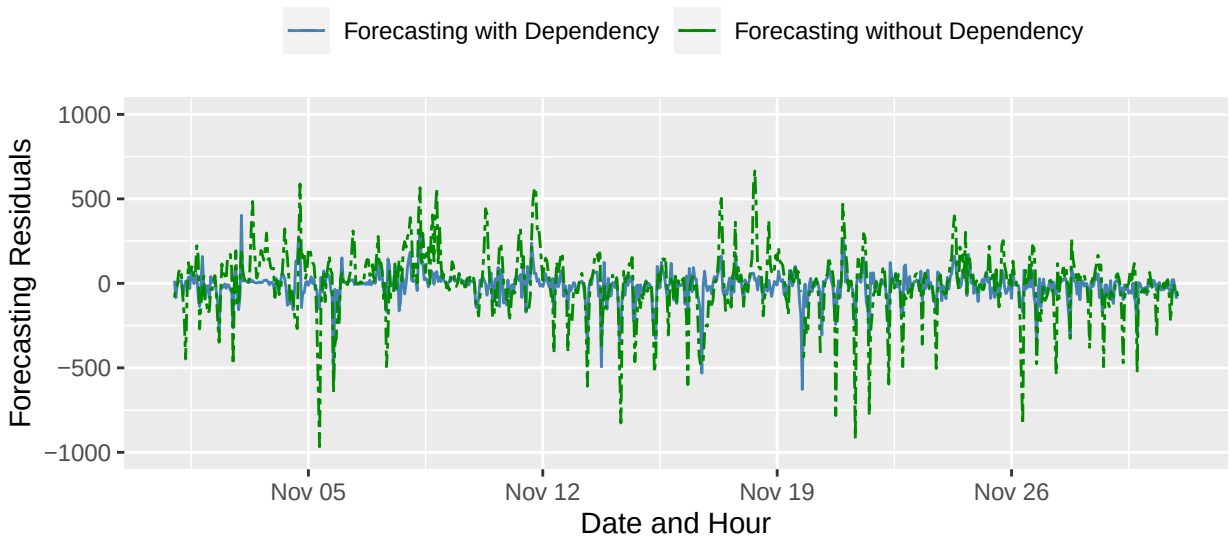

Figure 30: Forecasted Demand Comparison with Real-time Data Update

6 Conclusion

Our first contribution is that we redefined and explicitly stated the train data and the test data induced by the anchor date, in contrast to E., Park, and Cho 2020; E. and Cho 2020. Further, when considering different learning algorithms, Boosting performed the best when considering the original covariates, with an $R^2_{OOS} = 0.76$, which is lower than the $R^2_{OOS} = 0.92$ in E., Park, and Cho (2020). Being unable to replicate their R^2_{OOS} leads us to believe that they didn't consider an anchor date and did interleaving of their train and test data instead of having contiguous data, leading to leaking of the test data's distribution within the train data's distribution and thus artifically inflating the R^2_{OOS}.

Using Boosting as our learning algorithm, our second contribution is considering lagged supervisor values as additional covariates, i.e., incorporating the dependency of demand by adding additional predictor variables of the past hourly demand, which brings a significant improvement from using the original set of covariates without the dependency structure of the supervisor. Our work shows an improvement from 8.5 to 14.7 percentage points improvement in test R^2 when considering the correlation structure of bike rental demand.

And, finally, our third contribution is that we considered iterative statistical modeling in the context of two business scenarios, daily data update and real-time update. We showed that the data infrastructure capacity affects the forecasting results, e.g., more frequent data updates result in better prediction of bike rental demand.

A big limiting factor in our work has been the limited data we have had, with effectively one year of data. Our future work is to find similar bike sharing data but with a more extensive history. Then, we would like to impose our methodology and see whether we get similar results as shown in this study.

References

Berger, James O. (1985). *Statistical decision theory and Bayesian analysis*. New York: Springer-Verlag.

Box, George.E.P. and Gwilym M. Jenkins (1976). *Time Series Analysis: Forecasting and Control*. Holden-Day.

Breiman, L. et al. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks.

Breiman, Leo and E. Schapire (2001). "Random forests". In: *Machine Learning*, pp. 5–32.

Chen, Tianqi and Carlos Guestrin (2016). *XGBoost: A Scalable Tree Boosting System.* Tech. rep.

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml.

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

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

Efron, Bradley (2004). "The estimation of prediction error: Covariance penalties and cross-validation". In: *Journal of the American Statistical Association*, pp. 99–467.

Friedman, Jerome (1991). "Multivariate adaptive regression splines". In: *Annals of Statistics*.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (1998). "Additive Logistic Regression: a Statistical View of Boosting". In: *Annals of Statistics* 28, p. 2000.

Greene, William H. (2003). *Econometric Analysis*. Fifth. Pearson Education. URL: http://pages.stern.nyu.edu/~wgreene/Text/econometricanalysis.htm.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc.

Loh, W.-Y. (2011). "Classification and regression trees". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1, pp. 14–23. DOI: 10.1002/widm.8. URL: http://www.stat.wisc.edu/~loh/treeprogs/guide/wires11.pdf.

Russell, Stuart and Peter Norvig (2009). *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press.

Sheather, Simon (2009). *A Modern Approach to Regression with R*. Texts in Statistics. Springer.

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

Stone, Mervyn (1974). "Cross-validatory choice and assessment of statistical predictions". In: *Journal of the Royal Statistical Society: Series B*, pp. 111–147.

Therneau, Terry and Beth Atkinson (2019). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-15. URL: https://CRAN.R-project.org/package=rpart.

Zou, Hui and Trevor Hastie (2005). "Regularization and variable selection via the Elastic Net." In: *Journal of the Royal Statistical Society: Series B*, pp. 301–320.