CSE 780 - Final Project

Regression Analysis of Seoul Bike Sharing Dataset

Noah Frank
frankn1@mcmaster.ca

School Of Computational Science & Engineering
CSE 780: Data Science
Dr. Sharon McNicholas
McMaster University
April 7th, 2021

1 Data Description

1.1 Data Source

The data used for this project is the 'Seoul Bike Sharing Demand Data Set' retrieved from the UCI Machine Learning Repository.[14] The data set was provided to UCI from a study in which the authors combine the Seoul bike sharing demand data with weather data.[3] The resulting data set contains the number of bikes rented at each hour for a year, along with weather data and holiday data. In total, the data set is comprised of 13 features for each hour, which will be used to predict the number demand of rented bikes.

1.2 Variable Details

The data set is made up of 8760 observations with 13 features split between 8 numerical weather measurements, and 5 time features. The data set spans every hour from December 1st, 2017 to November 30th, 2018. Some preprocessing was performed to clean up the data set for better use in the analysis. One of the features 'functioning day' was used to remove all observations for hours in which the service was not running, and so the rental counts would all be zero. After removing these cases, leaving us with 8465 observations, the 'functioning day' feature was removed as all remaining observations would have the same value. Additionally, each observation was assigned a day of the week based on the date. The actual date was then removed from the data set before analysis, as each observation would have a unique date. There were no missing values for any of the observations. The resulting data set is then comprised of 12 variables. The final features used in the analysis are described further here:

- **Count:** Count of bikes rented at each hour
- **Hour:** Hour of the day
- **Temperature:** Temperature in degrees Celsius
- **Humidity:** Air humidity as a percentage
- **Wind Speed:** Current speed of wind in m/s
- **Visibility:** Meteorological visibility score reported in meters
- **Dew Point:** Dew point temperature in degrees Celsius
- **Solar Radiation:** Total solar radiation for the hour in MJ/m2
- **Rain:** Amount of liquid precipitation in mm
- **Snow:** Amount of solid precipitation in cm
- **Seasons:** Season of the year
- **Holiday:** Indicator of public holiday
- **Weekday:** Day of the week

1.3 Descriptive Analysis

The primary variable of interest in this analysis is the count of bikes rented at each hour, as this is the variable we will be attempting to model and predict. The distribution of the count variable can be seen in Fig. 1.

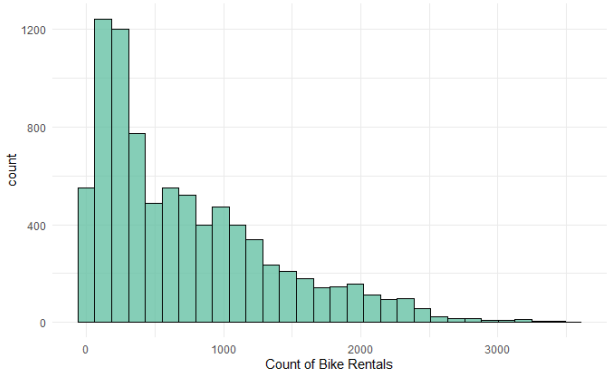

Figure 1: Histogram showing distribution of rented bike counts

From Fig. 1, we can see that the rental counts are skewed to the lower side of the scale. The count variable has a minimum of 2, mean of 729, and a maximum vale of 3556. In Fig. 2, we can see a time-series visualization of the rented bike count per day over the entire year of the data set. One can observe that the count begins low in the winter months, and rise throughout the spring to peak in the summer, and afterwards decreases back towards winter. Furthermore, at a glance we see less variance in the early data points of the winter months, compared to the later months having some of the highest daily counts, but also some of the lowest counts.

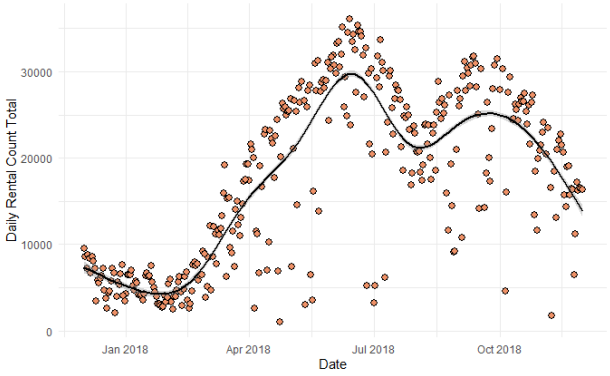

Figure 2: Daily rental count from Dec. 2017 to Nov. 2018

Next, we will look at the hour variable. This feature simply denotes the hour of the day when the observation occurs. Each observations occurs on a different and unique date-hour combination, with the hour being the smallest time step in the data set. In Fig. 3, we can see a box plot for the count variable at each hour. In this plot, we see that the count is lowest in the earliest hours of the day, and rise rapidly towards 8:00 AM. The counts are then steady again until rising in the afternoon, around 6:00 PM. These two peaks coincide very well with the commute time around a 9-5 job, which we will further inspect.

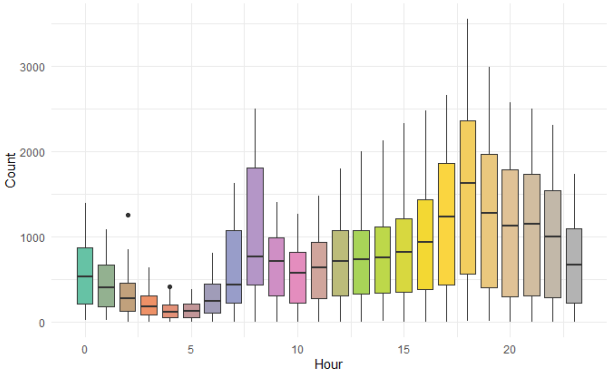

Figure 3: Box plot of bike rental count per hour

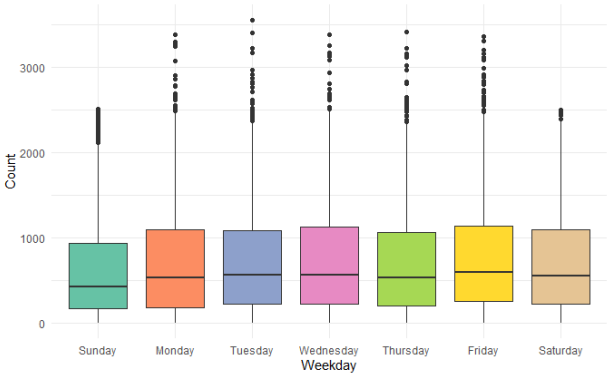

Figure 4: Box plot of bike rental count per weekday

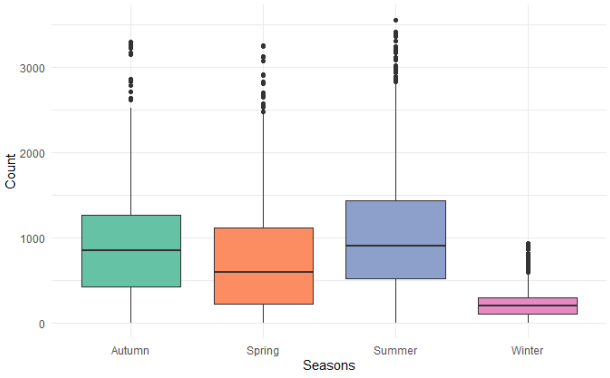

Figure 5: Box plot of bike rental count per season

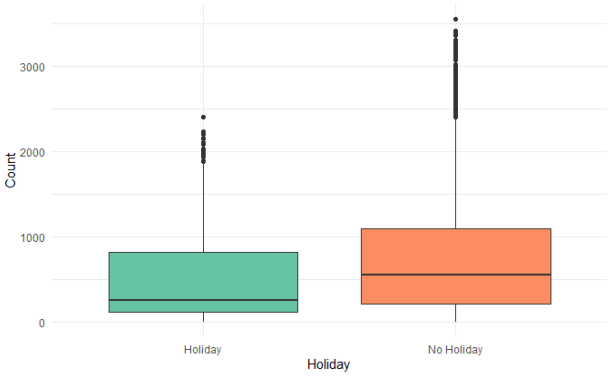

Figure 6: Box plot of bike rental count on holidays

Continuing from the hour feature, we can have a look at the other categorical features such as weekday, seasons, and holiday. The weekday feature gives the day of the week that each observation belongs to, ranging from Sunday to the next Saturday. As seen in Fig. 4, the box plots show that the average count per hour does not vary much from day to day, with the exception of slightly lower counts on Saturday and Sunday. The weekend also has less outliers, showing a smaller variance with smaller maximum count values. Another categorical variable is seasons, which is comprised of spring, summer, fall, and winter. From Fig. 5, we can see the highest median in summer, and the lowest median in winter. The values for winter are also much lower than any other month, giving a suitable feature to separate based on. Finally, the last categorical value to look at is holiday. This is a binary feature designating if a day is a public holiday, or if the day is a regular working day. As seen in Fig. 6, the observations taken during holidays have a lower median, and much less outliers than days that are not holidays.

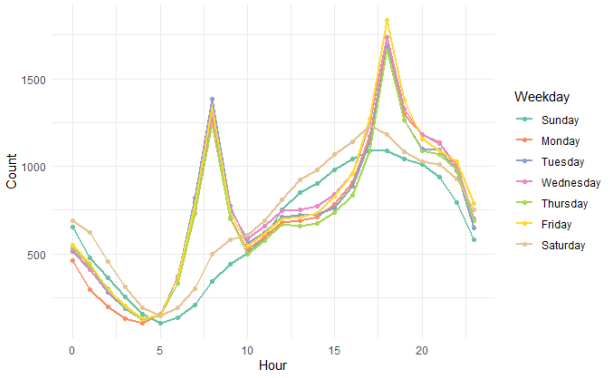

Figure 7: Hourly change in rental count per weekday

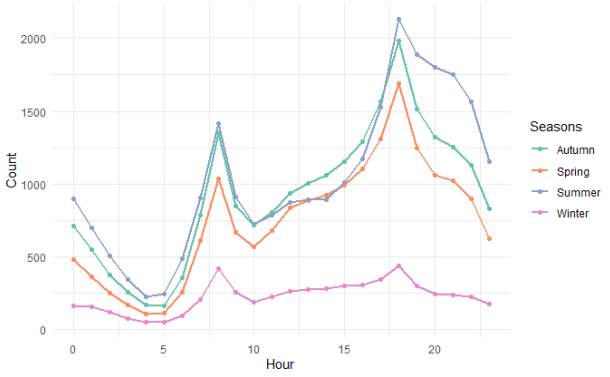

Figure 8: Hourly change in rental count per season

From Fig. 7, we can clearly see the relationship between the hour of day and the count during weekdays. There are noticeable peaks that occur during weekdays during commute times before and after usual work hours. This is even more evident when the same peaks are not visible for weekends. We also see a higher rental usage compared to baseline during working hours on the weekend, suggesting more time for

leisure as opposed to the shortened demand during work. In Fig. 8, we can see the hourly breakdown compared between the seasons. Again, the winter time shows the least amount of bike rental demand, but the commute peaks are still present. The remaining seasons are all considerably similar, with the exception of more demand after work hours on summer nights. The next set of variable to analyze are the weather features. As listed above, the weather variables include: temperature, humidity, wind speed, visibility, dew point, solar radiation, rain, and snow. Each observation is accompanied by these weather measurements taken at every hour. The first such measurement is temperature.

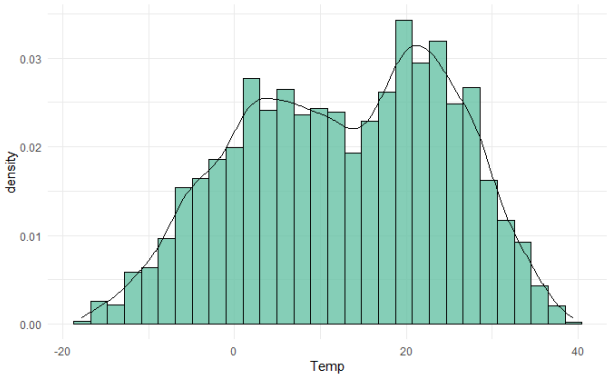

Figure 9: Density distribution of temperature variable

The temperature is a numerical variable measured in Celsius, with a distribution that can be seen in Fig. 9. The extremes of the temperature recording are a low of -17.8 degrees, and a high of 39.40 degrees. The mean temperature is 12.77 degrees. From the figure, we can see a slightly bi-modal distribution, which could be due to the variance in temperature caused by the seasons or the day-night cycle.

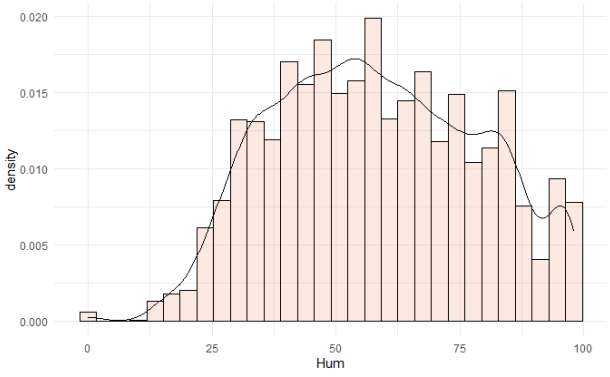

Figure 10: Density distribution of humidity variable

The next variable is humidity, another numerical variable measured as a percentage based on the saturation of water vapor in the air. This variable could take values from 0 to 100, but only the values from 0 to 98 are observed in this data set. The mean of the feature is 58.15, which can be seen from the distribution in Fig. 10.

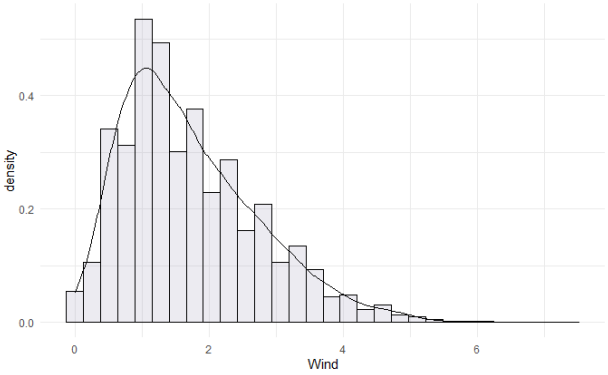

Figure 11: Density distribution of wind speed variable

Another one of the weather features is the wind speed numerical variable. This variable gives the maximum wind speed during the hour of observation, measured in meters per second. The distribution for this variable is shown in Fig. 11, with a Poisson-like shape. The variable takes on values from 0 to 7.4, with a mean of 1.726.

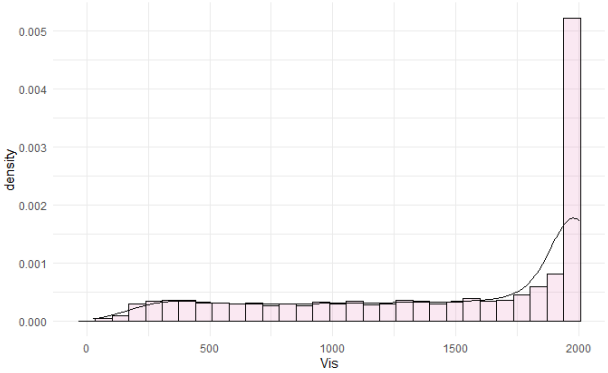

Figure 12: Density distribution of visibility variable

An additional weather feature is the visibility variable. This numerical variable gives the maximum visible distance, measured in meters. The distribution for this variable is shown in Fig. 12. The variable ranges from values from 27 to 2000, with a mean of 1434. As seen in the figure, the distribution is heavily skewed towards the upper values of visibility, showing that most observations occur in hours where there is no impairment to visibility due weather factors.

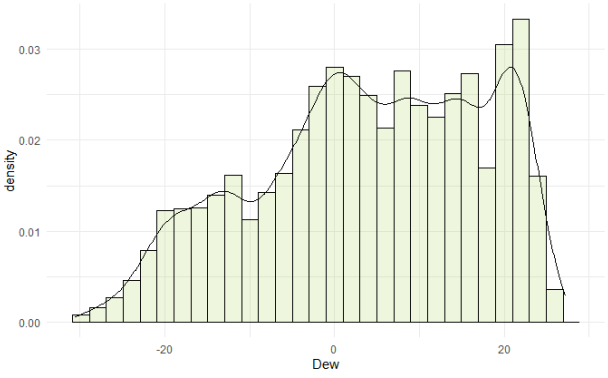

Figure 13: Density distribution of dew point variable

The dew point temperature is another numerical weather variable, denoting the temperature at which air must be cooled to saturate with water. This variable is closely linked with temperature and humidity. The distribution can be seen in Fig. 13, showing the minimum values of -30.6 and 27.2 degrees Celsius, respectively. The mean dew point temperature is found to be 3.95 degrees.

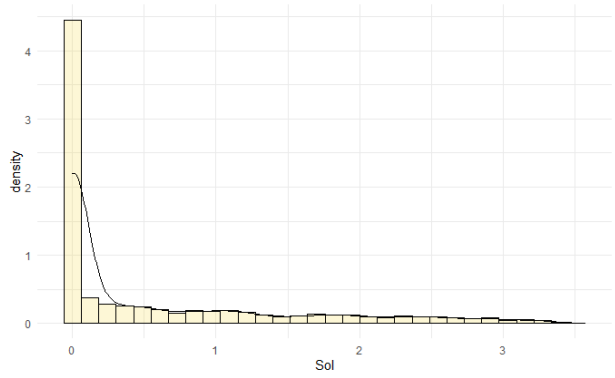

Figure 14: Density distribution of solar radiation

Solar radiation, is another weather measurement, giving a metric on the amount of energy due to the light emitted by the sun, measured in mega joules per meter squared. The distribution, shown in Fig. 14 has a minimum value of 0 and a maximum value of 3.52, with a mean of 0.568.

The final variables are for precipitation, giving the amount of rain and snow that have fallen in the recorded hour, in units of mm for rain, and cm for snow. The distributions for rain and snow can be seen in Fig. 15 and Fig. 16, respectively. Rain ranges from 0 to 35 with a mean of 0.15, whereas snow ranges from 0 to 8.8 with a mean of 0.08. From the figures, we can see that the median value for both of these variables is zero as the majority of observations have no rain, nor snow.

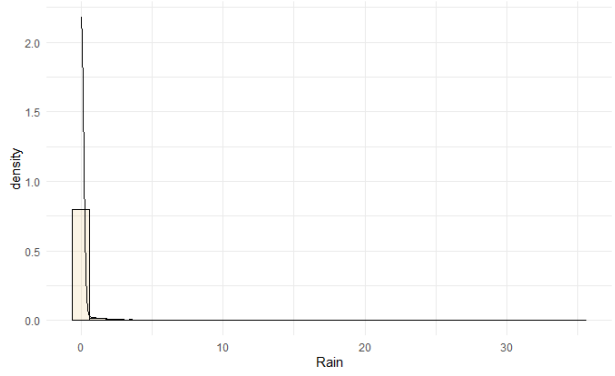

Figure 15: Density distribution of rain variable

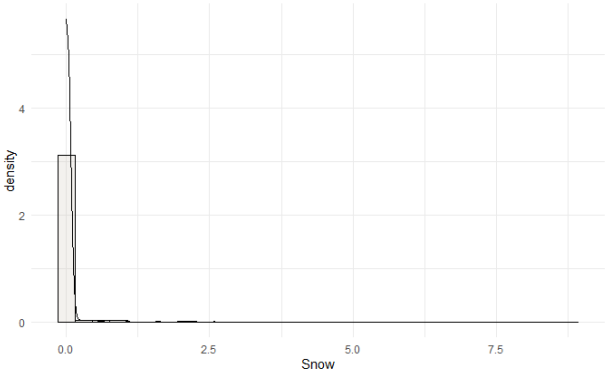

Figure 16: Density distribution of snow variable

Between the weather variables and the count of bike rentals, we can workout a correlation matrix to see which variables are most correlated with the others, as seen in Fig. 17. The strongest correlation is between the temperature and the dew point temperature, as temperature is a main factor for the dew point. We can then observe that the bike rental demand is positively correlated with temperature and dew point. The latter is likely due to the correlation between temperature and dew point. This positive correlation between count and temperature could be used as another strong predictive feature. In addition, we can see more about how the weather variables interact, seeing a negative correlation between humidity and visibility, wind and solar radiation.

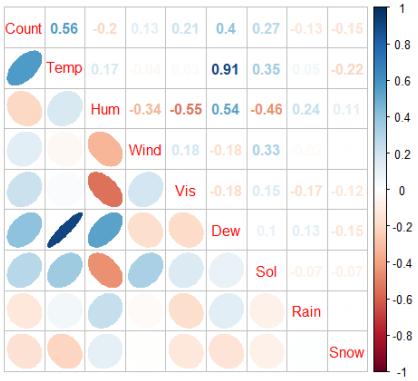

Figure 17: Correlation matrix between weather features and count

Another perspective from which to examine the weather variables are as a time series. In Fig. 18, we can see the mean of each weather variable per hour, separated by the season from which the observation belongs. This gives us a clearer picture of how the variables are affected by the time of day as well as the season. We can see some variables such as temperature, dew point and and snow which are well separated with respect to seasons. Additionally, we can see variables such as temperature, humidity, wind, visibility, solar radiation and snow which have well established cyclical patterns due to the time of day. We can also observe that rain seems to be random throughout the time of day as well as not having any well defined patterns across the seasons.

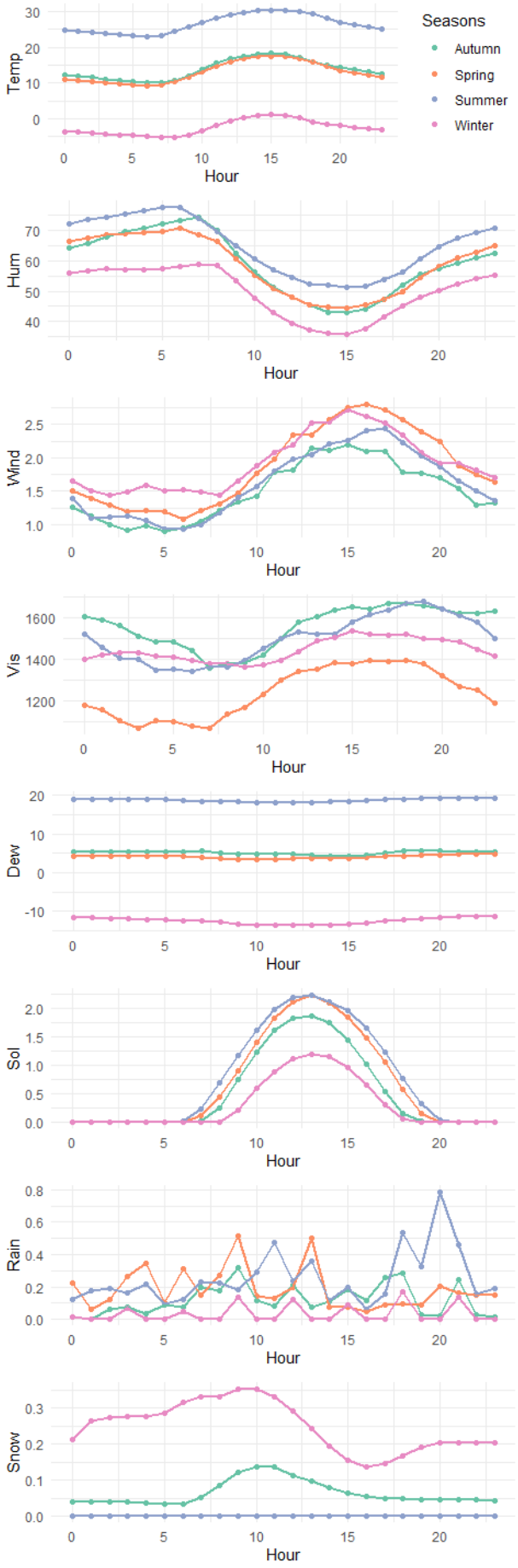

Figure 18: Hourly change in each weather feature by season

2 Problem Description

The aim of this analysis is to explore the Seoul bike sharing demand data set using a variety of analysis techniques for regression. The variable of interest in this data set is the count of bike rentals for each hour, and we will investigate the effect of each other feature in predicting our target variable. This data set uses simple and readily available features extracted from the local weather stations in addition to the rental companies own data. This type of data set could easily be used across many other similar businesses and industries that need to predict the demand for their services based on the time of the year and weather. Such information would be useful to have as it would allow for greater knowledge of demand leading to better management and decision making when it comes to supply. For this analysis, we will investigate the effectiveness of multiple regression techniques such as: Generalized Linear Models, k-Nearest Neighbours, Regression Trees, Bagging, Boosting, Random Forests, Projection Pursuit Regression and Neural Networks. The analysis will show which techniques are best suited to predict the bike demand based on the available features, as well as identify which features are the most important for this analysis problem.

3 Technique Description

3.1 Evaluation Metrics

We will be using multiple evaluation metrics in order to compare the performance of the different regression models presented. The performance metrics being used are: Root mean squared error (RMSE), mean absolute error (MAE), and R-squared (R^2). RMSE is given by the standard deviation of the residuals between the predicted values and the true observation values. Large errors can be identified using this measure and the fluctuation of model response regarding variance can be evaluated. Here the residuals are the distance between the predicted value and the truth value. The RMSE metric is a scale dependent metric, meaning that scaled down values must be scaled back up before computing the RMSE. RMSE can be given by the following equation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}}$$

R^2, also known as the coefficient of determination, gives a measure for the goodness of fit, ranging from values of 0 to 1. Higher values of R^2 mean that the predicted values better fit the observed values, up to a perfect fit at a value of 1. R^2 is calculated using the equation below:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \overline{Y}_i)^2}$$

MAE is another metric used to evaluate the magnitude of errors, without considering the direction of error. It is another scale-dependent metric, and must be treated accordingly when dealing with scaled down values. MAE is a linear metric, therefore all values are

weighted the same towards the total measure of error. The equation for MAE is given as:

$$MAE = \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{n}$$

Together, these metrics can be used to evaluate and compare the performance of the models. Training will be done with the goal of minimizing the target metric RMSE, whereas the evaluation of the final model ranking will compare the R^2 values.

3.2 Training Techniques

Cross validation is used when training the models in order to find the optimal model parameters through tuning. Initially, the data set is split into a training set and a testing set. The testing set is left aside until final model evaluation. The training set is split into k-folds of data, where k-1 folds are used to train the model and the remaining k fold is used to evaluate the model with the current parameters, this is called the validation set.[10] The process is repeated for k splits, so that each fold is held out in a certain iteration. The process can be seen visually in Fig. 19.

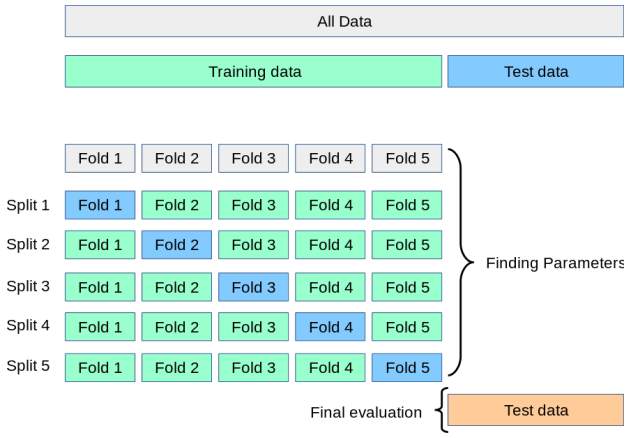

Figure 19: Example of 5 fold cross validation segmentation of data set[15]

After the cross-validation, the parameters are chose based on the performance of each training split. Using these final model parameters, we can re-fit the entire training data to produce the final model. The final model is then used to predict the target values from the testing set. The resulting predictions from the testing set are compared with the true values from the test data, which give us the evaluation metrics used to assess the final model performance based on data it has not been trained with. Holding out testing data is important to the assessment of the model as we can see how it behaves when exposed to data it has not previously seen. Overfitting to the training data can result in substantially lower evaluation metrics on test data. The goal is to find a model that learns well from training data, but also generalizes well to unseen testing data. Another method for preventing overfitting is to use bootstrapping when training the final model, so that it does not completely over fit to the entire training data.

3.3 Generalized Linear Models

The generalized linear model (GLM) is a type of linear regression in which the linear model is related to the response variable by a link function.[11] For this analysis we will be looking at the Gaussian family of generalized linear models, in which the link is the identity function: μ. That is saying that the expected value is related to the predictor by it's mean. The general linear model can be given by the equation:

$$\eta_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + ... + \beta_j x_{ji} + \epsilon_i$$

A link function is then used to describe how the mean depends on the linear prediction, given below:

$$E(Y_i) = \mu_i, g(\mu_i) = \eta_i$$

This leaves us with a multiple regression type model, where each variable is used to determine the effect on the predictor through a slope. The final model will then contain one value for each slope according to each variable, as well as one parameter giving the value of the intercept.

3.4 k-Nearest Neighbours

K-nearest neighbours (kNN) is non-parametric method that can be used for regression.[6] Using a value of k and a prediction point x_0, we can use kNN regression to identify the k closest observations relative to x_0, whose set is represented by $\mathcal{N}_0$. Using this set, the average is then calculated between the training points, to give the predicted value of the model with the given parameters, as given in the equation:

$$\hat{f}(x_0) = \frac{1}{k}\sum_{x_i \in \mathcal{N}_0} y_i$$

The value of the parameter k can be changed, which will change the predictions of the model. While training the model, a value of k will be determined based off of the evaluation metrics with regards to the training data. The optimal value for k will depend on the bias-variance trade-off.

3.5 Regression Trees

Regression trees are a form of decision tree model used to predict a continuous variable. The method for decision tree construction used here is known as the classification and regression tree approach.[1] These trees split the data into smaller groups and then fit a constant to each observation in the group. The grouping is achieved by recursive partitioning based on the different predictors. The value predicted is based on the average response values for all observations that fall are apart of a group. The model splits the data at internal nodes in the tree. At each node, the model partitions the data into two regions to minimize the overall sum of squares error as shown:

$$min(SSE = \sum_{i \in R_1}(y_i - c_1)^2 + \sum_{i \in R_2}(y_i - c_2)^2)$$

This process is recursively repeated until a stopping condition is met based off of a complexity parameter α. The complexity parameter penalizes the minimization function above resulting in:

$$min(SSE + \alpha T)$$

3.6 Bagging

Single regression trees have the drawback of having high variability between training data sets, and can easily over fit to the training data. One method of decreasing the variation and increasing the performance of the regression model is to use an ensemble method called bagging. Also called bootstrap aggregating, bagging uses bootstrapping to re-sample the entire training set with repetition, and fit a unique regression tree to each ensemble.[9] Combining the prediction from all the trees will lead to a regression estimate with less variance than the single regression tree, taking the majority vote of estimates. Bagging follows these steps: First, create m bootstrap samples from the training data, and train a single regression tree for each bootstrap sample. Next, average each individual prediction from each tree to create an overall predicted value for the bagging model. A free parameter that can be tuned with the bagging model is the number of trees to average over.

3.7 Random Forests

Another ensemble method is called random forests. This is a modified version of bagging in which we do not consider all predictors from the data set at every split in the tree. This is used as another method to diversify the trees by forcing the splitting to choose from different variables, as opposed to always resolving the same tree based on the same splitting rules. For each tree, at each split, a different number of predictors, $\mathcal{M}$, is considered. The random forest method can be adjusted by tuning the number of trees and the number of predictors at each split. A value for $\mathcal{M}$ often used is: $\mathcal{M} = \sqrt{p}$, where p is the number of predictors. Like in the bagging method, a number of trees can be selected, for which each tree will be a unique model based on the random predictors available at each split. Averaging all the predictions for each tree will result in the overall prediction from the random forest.

3.8 Boosting

The final tree ensemble method to explore is boosting, also known as gradient machine boosting (GBM), or multiple additive regression trees (MART). In boosting, the regression trees are grown sequentially with respect to each other, rather than independently in parallel and then combined, as is the case for bagging. The trees are sequentially linked, meaning the output of the first tree will be the input to the second tree, and so on. The goal of boosting is to pass on the tree's output and a weight relative to it's accuracy in order to ensemble a weighted sum of these measures. The end product is to minimize the overall error according to the weights and residuals passed through the model in an iterative manner. Overall, boosting takes multiple weak learners, such as individual regression trees, and can form a strong learner. This method can be adjusted by tuning certain parameters including: The number of trees, the shrinkage parameter, and the depth of each tree. Boosting can over fit the model to the training data if the number of trees is too large.

3.9 Projection Pursuit Regression

Projection pursuit regression (PPR) is a statistical model that projects the explanatory variables in the optimal direction before applying smoothing functions.[5] The model is used to predict a target Y based on p-dimensional X, using linear combinations of ridge functions. If ω_m is a unit p-vector, the basic PPR model takes the form:

$$f(X) = \sum_{m=1}^{M} g_m(\omega_m^T X)$$

The function $g_m(\omega_m^T X)$ is called a ridge function, and varies only in the direction along the vector ω_m. The goal of PPR is to compute the projection of the estimating parameters X onto the unit vector ω_m, seeking a good fit.[13] The PPR model is fit by minimizing the following error function:

$$\sum_{i=1}^{N} [y_i - \sum_{m=1}^{M} g_m(\omega_m^T x_i)]^2$$

For this model, the value for M needs to be chose, defining the amount of ridge functions to include in the fitting process. PPR is a useful model for prediction, however as M increases, the interpretation of the fitted model becomes more difficult.

3.10 Neural Networks

Neural networks are another type of non-linear statistical model. In this analysis we will be looking at a single hidden layer neural network, as displayed in Fig. 20.

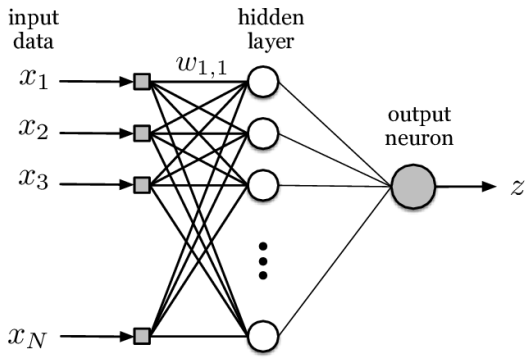

Figure 20: Example neural network architecture[2]

This neural network is a two stage regression model, with a single output representing the predicted value of the model. The hidden layer is composed of nodes which receive a weighted input from each input parameter of the data set, in addition to a bias. The target node, also known as an output neuron then compute a linear combination of each hidden layer. The neural network can then be described by the following equations, where X denotes the input data, Z denotes the hidden layer, and T is the output[12]:

$$Z_m = \sigma(\alpha_{0m} + \alpha'_m X),$$
$$T_k = \beta_0 k + \beta'_k Z,$$
$$f_k(X) = g_k(T)$$

The α and β values represent the weights and biases at each layer. Another function, σ, represents the activation function of the neural network, with common functions being the sigmoid function, or the ReLU function. The neural network is fit to the training data according to back propagation and gradient descent.[5] At each pass with the training data, the network is fixed to the current weights and the predicted values are computed, which is then used to calculate error in order to adjust the gradient descent.

4 Result Description

For all analyses, the data was randomly split into a training and testing set with 75% of the data in the train set, and 25% of the data in the test set. This split leaves us with 6348 observations of training data points, and 2117 observations of testing data points. From the training set, each model is trained using 10 fold cross validation to establish the optimal tuning parameters for each model. In addition, once the tuning parameters have been determined, the models are fit to the entire training data set using 25 repetition bootstrapping in order to improve model stability reduce overfitting to the training data, as is the default in the 'caret'[7] package used to fit the models in R. Once the final model has been established, it is used to predict bike rental counts for the entire test data set. These predictions are then compared against the truths to give our model metrics of RMSE, MAE, and R^2.

4.1 Generalized Linear Models

The first model used was the GLM, using the Gaussian family, which is the same as a linear regression model. The 'glm' function in R was used with the training functionality of 'caret'. There were no tunable parameters in this model, and the final parameters are given in Table 1.

Table 1: Final Model Parameters

Variable	Estimate	Std. Error	p-value
(Intercept)	738.622	5.503	<2e-16
Hour	202.014	6.073	<2e-16
Temp	184.994	52.560	0.000435
Hum	-240.931	25.002	<2e-16
Wind	17.476	6.282	0.005420
Vis	4.680	7.203	0.515843
Dew	178.677	60.156	0.002987
Sol	-67.355	7.819	<2e-16
Rain	-70.918	5.721	<2e-16
Snow	13.773	5.823	0.018053
Spring	-61.791	7.209	<2e-16
Summer	-75.844	8.969	<2e-16
Winter	-155.550	10.157	<2e-16
Holiday	24.540	5.581	1.12e-05
Monday	29.243	7.248	5.53e-05
Tuesday	33.955	7.163	2.18e-06
Wednesday	46.570	7.232	1.29e-10
Thursday	33.788	7.221	2.94e-06
Friday	42.017	7.210	5.90e-09
Saturday	22.464	7.231	0.001901

The performance of the model on the training and testing data can be seen in Table 2. below. We can see that the error is still relatively large and the R^2 value is not very strong.

Table 2: Final GLM Model Metrics

Data set	RMSE	MAE	R^2
Training	440.747	328.9613	0.5434576
Testing	416.782	318.2528	0.5491056

4.2 k-Nearest Neighbours

The next model used was the kNN model. The 'knn' method was used within the training function of 'caret' in R. The tunable parameter in this model is k, the number of neighbours to consider when predicting a new value. In Fig. 21 we can see the training error for the cross validation models as the k parameter is altered. An optimal value of k = 4 was chosen as it minimizes the target training metric RMSE. A value of 3 could have been chosen as it was nearly the same error as in the value of 4 for k. The performance of the final model on the training and testing data can be seen in Table 3.

Table 3: Final kNN Model Metrics

Data set	RMSE	MAE	R^2
Training	363.3467	235.8254	0.6963239
Testing	308.4763	200.1129	0.7535775

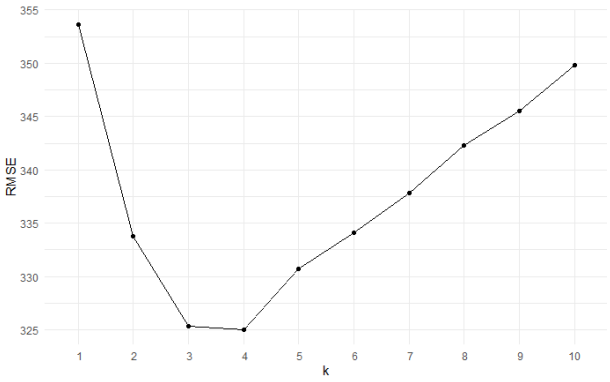

Figure 21: kNN parameter tuning: k vs. RMSE

4.3 Regression Trees

Another model used is the regression tree model, a form of decision tree. The 'rpart'[16] package was used within the training function of 'caret' in R. This model must be tuned to select for a value of 'cp', the complexity parameter which determine the depth of the regression tree. In Fig. 22 we can see the training error for the cross validation models as the cp parameter is modified. An optimal value of cp = 2.5e-6 was chosen as it minimizes RMSE, the target metric for training. The performance of the final regression tree model on the training and testing data can be seen in Table 4.

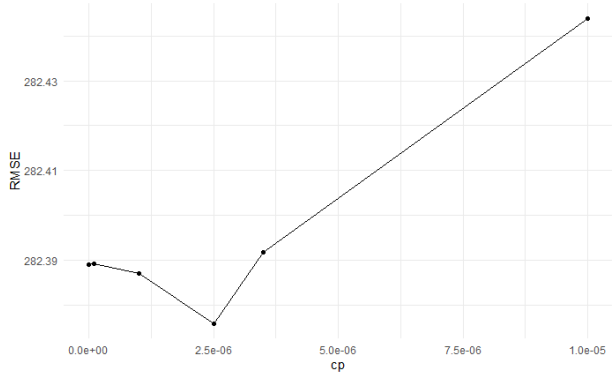

Figure 22: CART parameter tuning: cp vs. RMSE

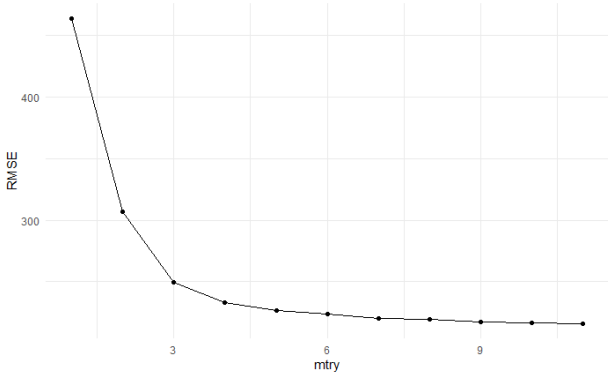

Figure 23: RF parameter tuning: mtry vs. RMSE

Table 4: Final CART Model Metrics

Data set	RMSE	MAE	R^2
Training	294.9239	200.9711	0.8005984
Testing	268.1104	162.1561	0.8134643

4.4 Bagging

The bagging model, was used with the 'randomForest'[8] package within the training function of 'caret' in R. For bagging, we use a value of mtry equal to all the predicting features, which is 12 in the case for this data set. This model was also tuned to select for a number of trees, which was determined to be 500, which was the lowest amount of trees per model giving the same relative RMSE. The performance of the final bagging regression tree model on the data sets can be seen in Table 5.

Table 5: Final Bagging Model Metrics

Data set	RMSE	MAE	R^2
Training	226.8345	139.4788	0.8794354
Testing	203.3471	126.6508	0.8925242

4.5 Random Forests

Expanding on bagging, a random forest model was made using the 'randomForest'[8] package called through the 'caret' training function. For random forests, we iterate through each possible value of mtry. In this case values of 1 to 11 were compared and the error at each value can be seen in Fig. 23. The value of 11 was chosen, as we can see that the more variables considered at each split lead to less error. This model was also tuned to select for a number of trees, which was determined to be 500, the same number as with the bagging model. The performance of the final random forest model can be seen in Table 6.

Table 6: Final Random Forest Model Metrics

Data set	RMSE	MAE	R^2
Training	226.9735	139.8242	0.87939
Testing	203.9827	127.369	0.8919308

4.6 Boosting

The next model is boosting, which was implemented using the 'gbm'[4] package as a method within the 'caret' package training function. Through parameter tuning, the optimal interaction depth was determined to be 10, and the shrinkage was found to be 0.1. In order to prevent overfitting the number of trees was determined to be 145, where the training deviance begins to flatten out, as seen in Fig. 24. The final model's performance can be seen in Table 7.

Table 7: Final Boosting Model Metrics

Data set	RMSE	MAE	R^2
Training	194.8859	124.8467	0.9107172
Testing	177.3508	112.5599	0.9177456

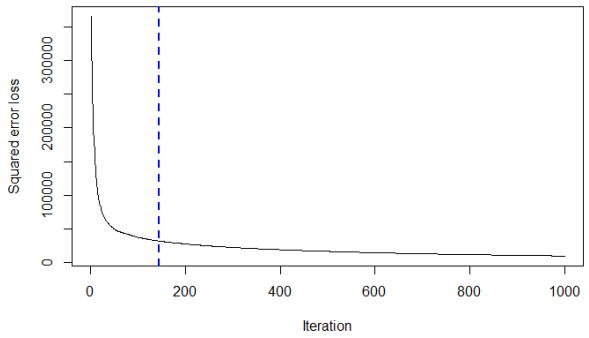

Figure 24: Boosting parameter tuning: ntrees vs. RMSE

4.7 Projection Pursuit Regression

Another model is projection pursuit regression, implemented through 'caret' using the 'ppr' method of the train function. The tunable parameter for this model is 'nterms', the number of terms to be included in the final model. In Fig. 25, we can see the error as terms are added on. An optimal value of 17 was chosen as it

is the lowest number of terms with minimal error. The results of the final model are shown in Table 8.

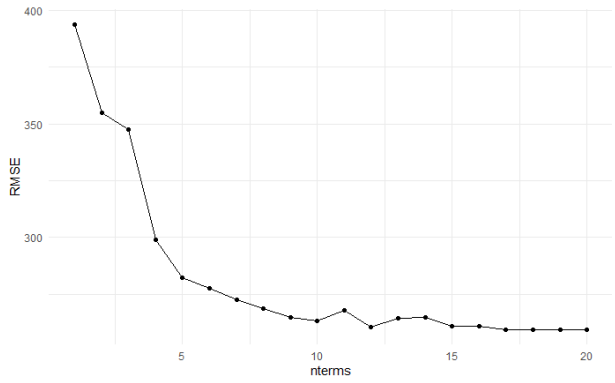

Figure 25: PPR parameter tuning: nterms vs. RMSE

Table 8: Final PPR Model Metrics

Data set	RMSE	MAE	R^2
Training	283.1152	200.9711	0.810956
Testing	235.8429	166.6251	0.8546334

4.8 Neural Networks

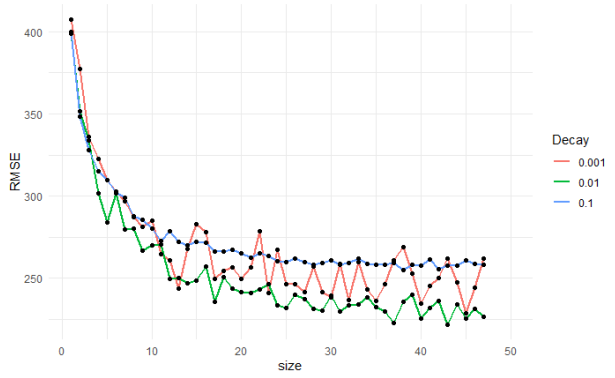

Figure 26: NNet parameter tuning: size vs. RMSE per decay rate

The final model to investigate is the single layer neural network. This model is implemented by the 'nnet'[17] package in the 'caret' train function. The parameters to tune in this model are the number of neurons in the layer of the neural network, as well as the decay rate. As seen in Fig. 26, the optimal combination of these parameters is a size of 43 and a decay value of 0.01. The performance of the neural network can be seen in Table 9.

Table 9: Final NNet Model Metrics

Data set	RMSE	MAE	R^2
Training	213.5059	145.0022	0.8543293
Testing	229.3033	155.0285	0.8631765

4.9 Model Comparisons

From each individual model we have looked at, we can compare the training and testing metrics as seen in Table 10 and Table 11, respectively. The table shows the models ranked from smallest error to largest error, with the best model at the top. We can see that the boosting model had the best performance on the test data set with an R^2 value of 0.92, showing a pretty good prediction ability. We can also the the variable importance plots of our 3 strongest models in Fig. 27. From all three models we can see that temperature and hour of the day are the 2 most important features across all models. We can also see important weighting on some of the other weather measurements such as solar radiation, humidity and rain. Distinguishing between the days of the week did not carry a very strong importance to the analysis, however the winter season was a quite important variable in the model. The presence of holidays, as well as snow did not play a major factor in the variable importance, as these were more rare events and were not very prevalent in the data set.

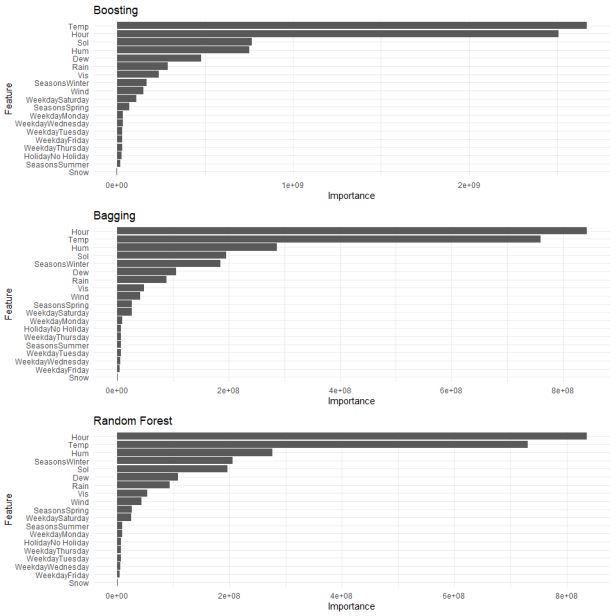

Figure 27: Ensemble variable importance plots

Table 10: Final Model Training Metrics

Models	RMSE	MAE	R^2
Boosting	194.8859	124.8467	0.9107172
Bagging	226.8345	139.4788	0.8794354
RF	226.9735	139.8242	0.87939
NNet	213.5059	145.0022	0.8543293
PPR	283.1152	200.9711	0.810956
CART	294.9239	200.9711	0.8005984
kNN	363.3467	235.8254	0.6963239
GLM	440.747	328.9613	0.5434576

Table 11: Final Model Testing Metrics

Models	RMSE	MAE	R^2
Boosting	177.3508	112.5599	0.9177456
Bagging	203.3471	126.6508	0.8925242
RF	203.9827	127.369	0.8919308
NNet	229.3033	155.0285	0.8631765
PPR	235.8429	166.6251	0.8546334
CART	268.1104	162.1561	0.8134643
kNN	308.4763	200.1129	0.7535775
GLM	416.782	318.2528	0.5491056

In Fig. 28, we can visually inspect the quality of fit of the model, showing the predicted values of each model on the horizontal axis, with the true value of each observation on the vertical axis. As we can see, the models with better performance fit this linear relationship between the the true value and predicted value, whereas some of the worse models do not show this relationship. Between the models with the linear relationship, those with smaller residuals have less error and a better R^2 score.

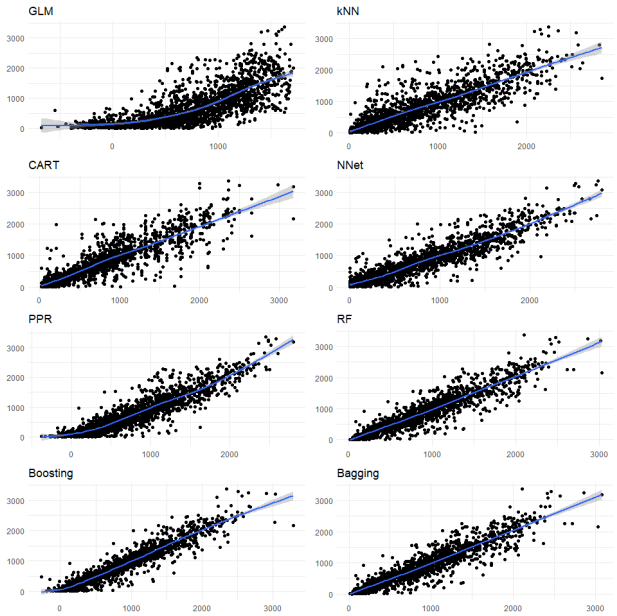

Figure 28: Predicted values vs. truth for all models

5 Conclusions

The regression analysis carried out with different models showed a variety of responses between the models. From the results we have seen that boosting was the best model to predict the bike rental demand, followed closely by bagging and random forests. We can clearly see the strength of ensemble methods of regression trees and their effectiveness with this data set. With an RMSE of 177.3508, a MAE of 112.5599, and an R^2 score of 0.9177456, the boosting model shows desirable results. From the variable importance plots, we can also see how each of the collected features contributes to the final model. As we saw from the correlation matrix in Fig. 17, the temperature had a moderate correlation with the count of rented bikes, giving us an indication that this would be an important variable. We can also see how the hour of the day was another greatly important factor. This fortifies the idea that much of the bike rental demand is for commuters getting to work in the morning, and back home in the evenings. This was seen clearly in Fig. 7 of the exploratory analysis, as the rush hour peaks in demand were not present in the weekend. This shows us that knowledge of the day of the week, as well as the hour of the day can give a large amount of information about the demand for the bike sharing service. This analysis showed us the effectiveness of various regression models on the data set and shows potential benefits for similar problems in this industry. For bike sharing services to ride sharing services, it is important to be able to predict demand in order to better manage supply and available resources. This analysis can be extended to ride sharing services, by including location data in order to estimate the demand for a driver, so that the service could better manage their fleet of drivers. The importance for this type of model can not be overstated as some industries are seen shifting towards gig-economies where companies must manage groups of in dependant drivers and delivery people. Further work on this analysis could be conducted by including location data to predict the demand based on location.

References

[1] *Classification and regression trees*. Repr. Chapman & Hall [u.a.], 1998. ISBN: 9780412048418.

[2] Lucas Cuadra et al. "Computational intelligence in wave energy: Comprehensive review and case study". In: *Renewable and Sustainable Energy Reviews* 58 (May 2016), pp. 1223–1246. DOI: 10.1016/j.rser.2015.12.253.

[3] 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* 53.sup1 (2020), pp. 166–183. DOI: 10.1080/22797254.2020.1725789.

[4] Brandon Greenwell et al. *gbm: Generalized Boosted Regression Models*. R package version 2.1.8. 2020. URL: https://CRAN.R-project.org/package=gbm.

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

[6] Gareth James et al. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. URL: https://faculty.marshall.usc.edu/gareth-james/ISL/.

[7] Max Kuhn. *caret: Classification and Regression Training*. R package version 6.0-86. 2020. URL: https://CRAN.R-project.org/package=caret.

[8] Andy Liaw and Matthew Wiener. "Classification and Regression by randomForest". In: *R News* 2.3 (2002), pp. 18–22. URL: https://CRAN.R-project.org/doc/Rnews/.

[9] Prof. McNicholas. *CSE780: Bagging, Random Forests, and Boosting*. McMaster University, School of Computational Science & Engineering. 2021.

[10] Prof. McNicholas. *CSE780: Cross-Validation*. McMaster University, School of Computational Science & Engineering. 2021.

[11] Prof. McNicholas. *CSE780: Cross-Validation*. McMaster University, School of Computational Science & Engineering. 2021.

[12] Prof. McNicholas. *CSE780: Neural Networks*. McMaster University, School of Computational Science & Engineering. 2021.

[13] Prof. McNicholas. *CSE780: Projection Pursuit Regression*. McMaster University, School of Computational Science & Engineering. 2021.

[14] UCI Machine Learning Repository. *Seoul Bike Sharing Demand Data Set*. 2020. URL: https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand.

[15] Scikit Learn. *Cross-validation: evaluating estimator performance*. [Online; accessed April 5, 2021]. 2020. URL: https://scikit-learn.org/stable/modules/cross_validation.html.

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

[17] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Fourth. ISBN 0-387-95457-0. New York: Springer, 2002. URL: http://www.stats.ox.ac.uk/pub/MASS4.