## Pre-processing and Exploratory data analysis

Pre-processing

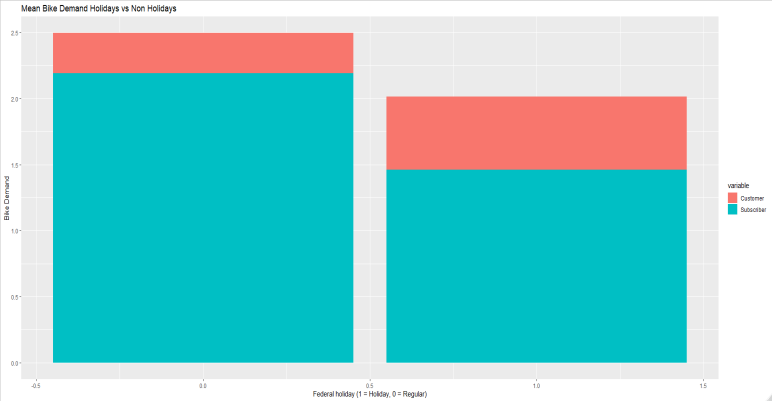
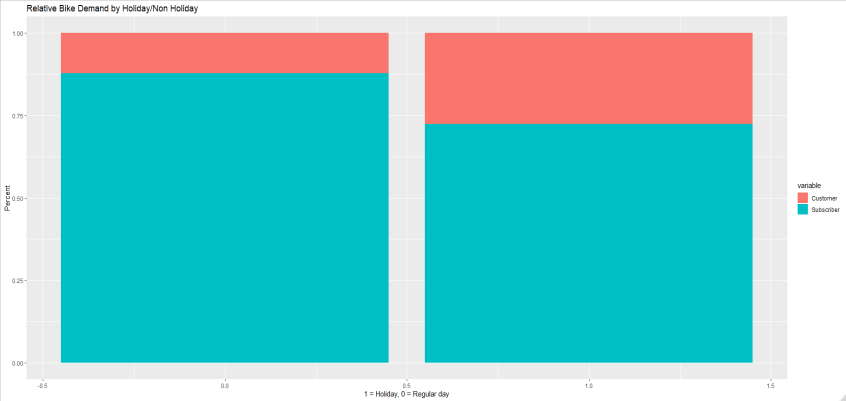
* Extracting the year, month, day, and hour variables from the start.time in the trip\_data.
* Removing the trip.id, End.Station and End.Data variables from the trip\_data.
* Create the Total Variable in the trip\_data.
* Counting the Customers / Subscribers from the trip\_data in a separate df.
* Adding the federal holidays from <https://www.officeholidays.com/countries/usa/2019>.
* Adding the weekdays in the trip\_data.
* Adding the zip code for the station\_data.
* Create dummy variables for the Events in the weather\_data.
* Imputing the missing values in the weather\_data.
* Merge the 4 dataframes (trip\_data, weather\_data, station\_data, user\_data).
* Sort the dataframe by year, month, day, and hour.

### Exploratory data analysis

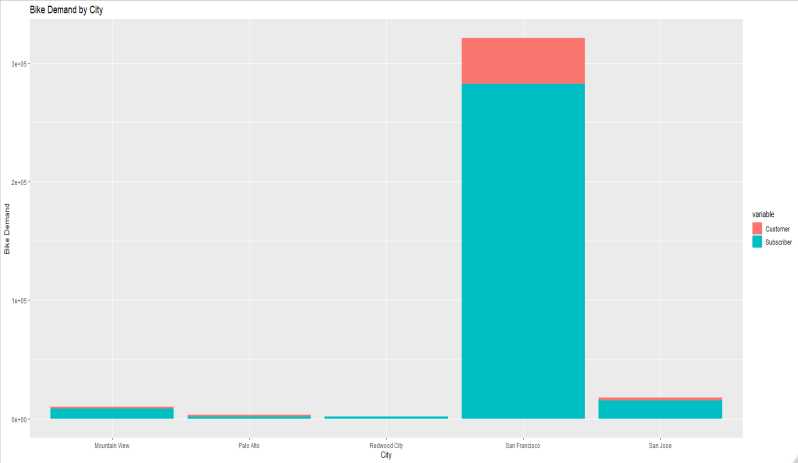
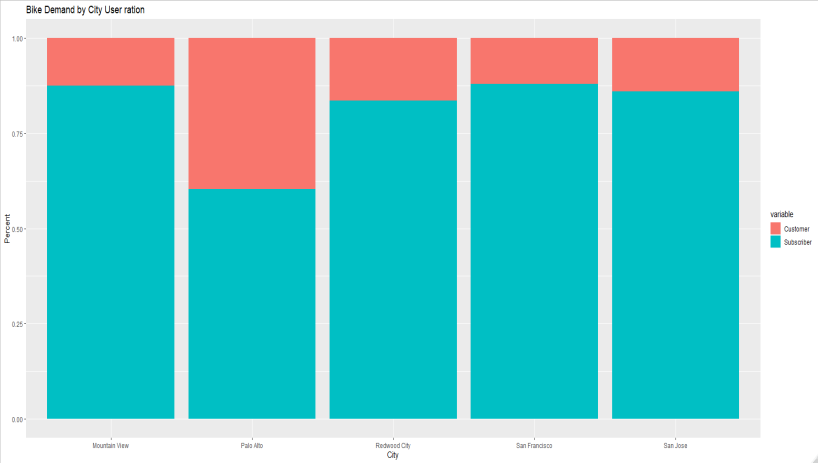
There are stations that have high demand, while others that barely have any. The ratio of Customer type users to Subscriber type users also vary by station.

## 

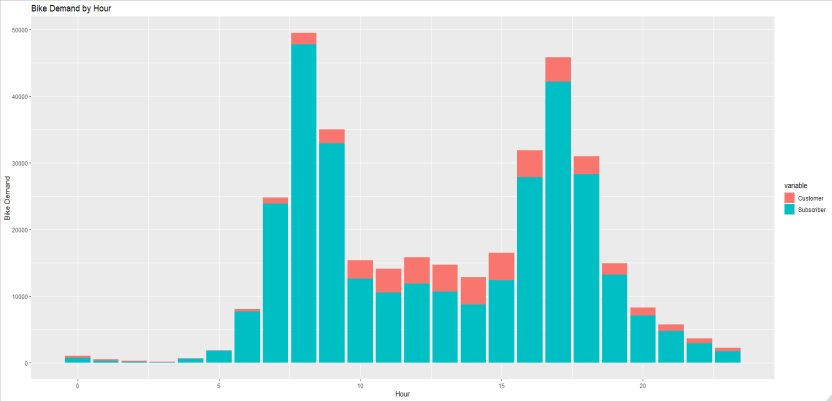
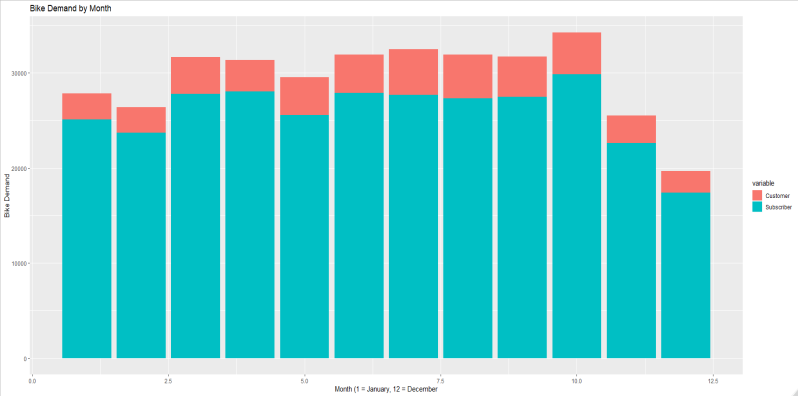
During federal public holidays, the average number of bikes demanded by station and hour is lower than on normal days. The Customer to Subscriber ratio is however higher.

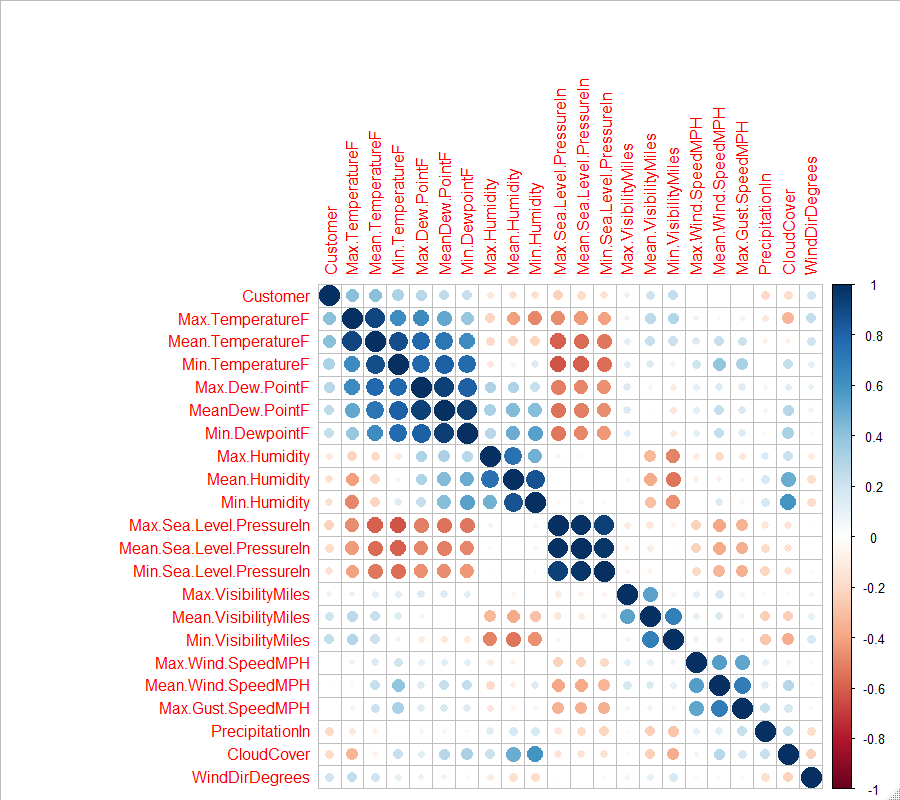
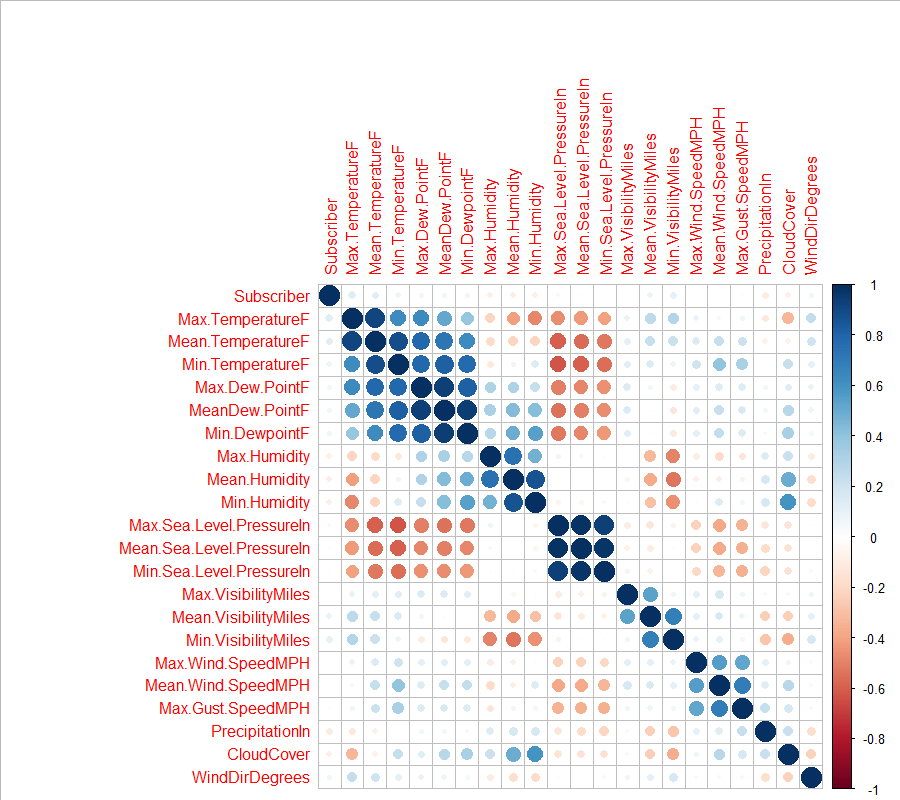
The demand for bikes in San Francisco is the highest, while the Customer to Subscriber ratio in Palo Alto is the highest. Both of these can be considered outliers.

The user base is smaller during the winter month than during the rest of the year. At the same time, the demand of the Subscriber user base peaks between 7 and 9, and between 16 and 18, whiles the Customers’ between 10 and 18.

****

The correlation between the weather variables for the Subscriber type demand (left) is low, which means that people who are subscribed are not that influenced by the weather when using the service. However the correlation for the Customer type users (right) is moderate, which means that this user group is influenced by the weather to some degree.



## Modeling Approach

### Model Selection

Considering that the predictive variables don’t have normal distributions due to the cyclical nature of the variables (hours, weeks, months, and years), a linear model would not perform well on this dataset. Furthermore, since the data points had been collected at irregular timeframes from different stations, time series forecasting would not be an efficient choice. Based on these characteristics of the data the two remaining algorithm options would be a tree based algorithm or a neural networks. Neural networks would require more data for a good performance. From the tree based algorithms, the most popular ones are random forest and xgboost. Out of these two, xgboost has too many hyper parameters that need to be tuned considering the resources available, while random forest only has 2. Therefore the logical choice would be to use the random forest algorithm using the randomForest library.

### Feature selection

Since the company needs to predict the demand for a specific station at a specific hour on a specific day, the variables: day, month, year, hour, station must be included in the final model.

The features with duplicate information values were removed:

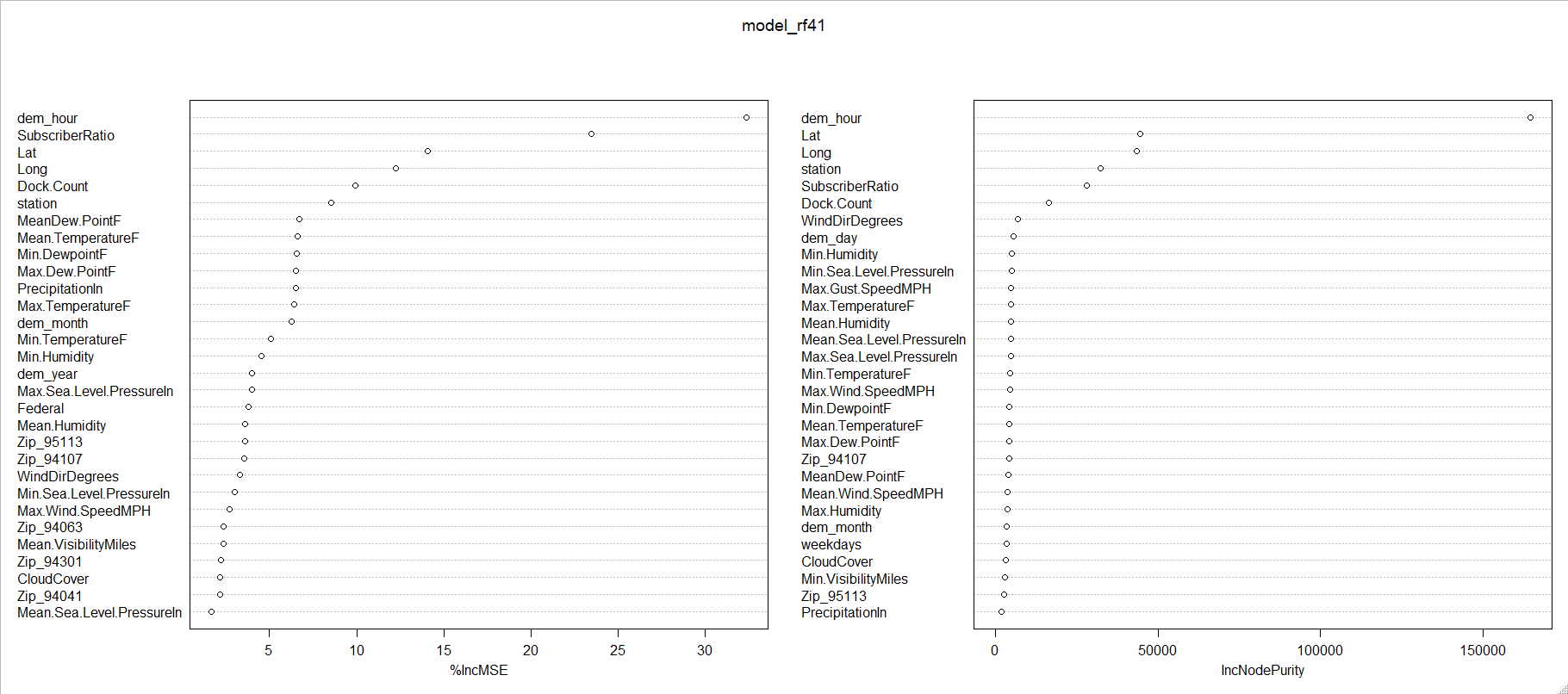
* The name of the station, which has a 1 to 1 relationship with the stations id, while the station id also distinguishes between locations with the same name but with different location.
* The city name, which has a 1 to 1 relationship with the zip code.
* The Start.Date, which had been broken down into year, month, day, hour.

When selecting feature for the model, the variables selected also need to be available when the customer would like to use the model to predict future demand. Most of the features used in the model are readily available, for example: the information related to each station, the geographical data and the date/time, while the weather data can be collected from forecasting websites. The percent of Subscribers in the total demand however is not known in advance. Therefore in a production model this would need to be omitted (which would hurt the model’s performance), or a best estimate must be used based on the historical data. Besides the Subscriber percent, the destination stations and the average trip length could also make the model more accurate. However these would be difficult to estimate before a trip occurred, and because of this, these would be omitted.

Unfortunately tuning the randomforest mtry parameter was not feasible due to resource constraints. Therefore the rounded square root of the number of variable used in each forest was used as the value for mtry. The data frame was also sorted by year, month, day, hour, in order to prevent data leakage, and the first 90% was used as training data, while the remaining 10% as testing data.

After training an initial randomForest algorithm with all the remaining variables, the test MSE stabilized around 3.03 without replacement, and around 3.04 with replacement. Next, the data for the moved stations were removed, and another randomForest was trained using the same parameters. At the end the test MSE was: 3.05, which show that the model does not perform better with the trip data for the outdated station ids excluded. In each case the ntree = 100.

The variable importance plot shows that using a subset of the features could lead to better results.



Using only the features with a %IncMSE score and using 80 trees per model:

* above 0.05, with 27 features in total, resulted in a test MSE of 2.85
* above 0.1075, with 21 features in total, resulted in a test MSE of 2.82
* above 0.11, with 17 features in total, resulted in a test MSE of 2.72
* above 0.1125, with 15 features in total, resulted in a test MSE of 2.88
* above 0.12, with 12 features in total, resulted in a test MSE of 2.72
* above 0.15, with 11 features in total, resulted in a test MSE of 2.69

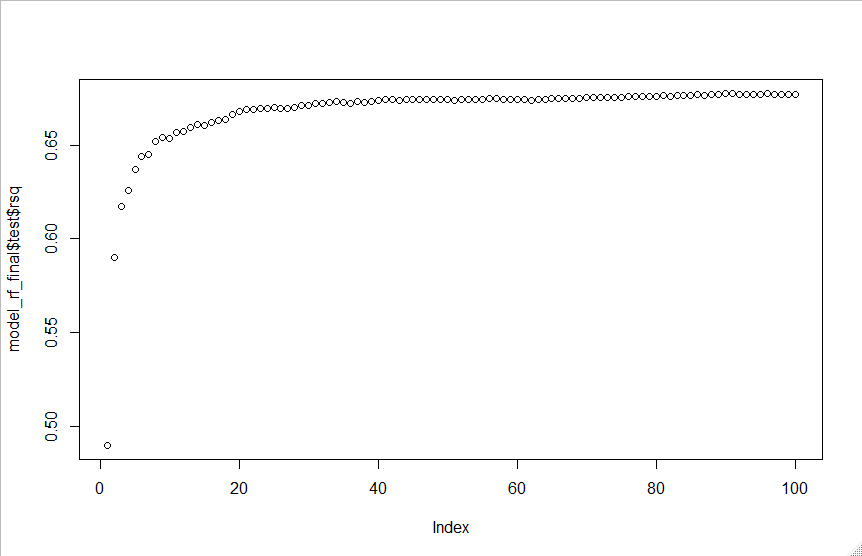
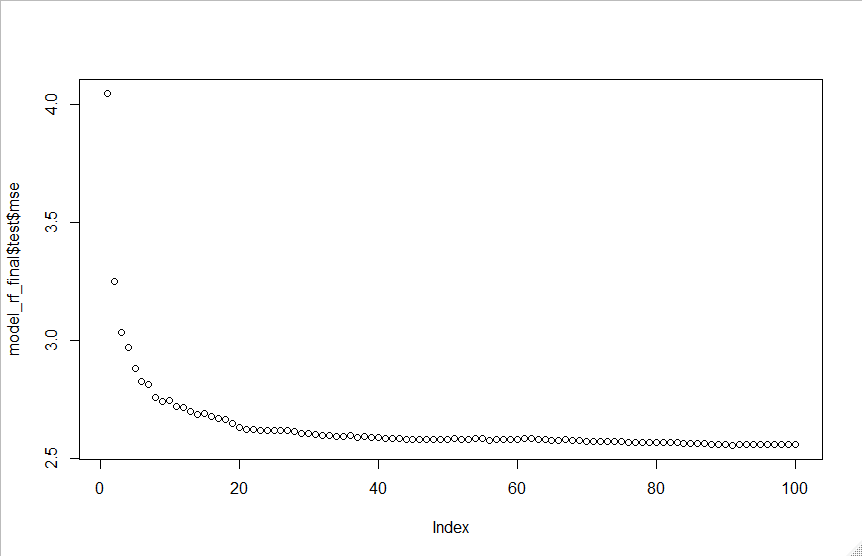
After trying out different subsets of features, there are several subsets of features that produce near equal results. Since the results of these are nearly identical, the model with the least number of predictors (11) would be chosen.

### Model tuning

After finding the optimal “mtry” parameter for the selected features using the tuneRF function of the randomForest package, a final randomForest algorithm was trained using the first 90% as training data, and the remaining 10% as testing data. The final model parameters: mtry = 6, ntrees = 100, replacement = False.

### Model performance:

The performance of the final model stabilized around 50 trees with a test MSE of 2.56 (left) and with 67.78% of the variance explained (right).



The two error measures of the final model that is worth looking at are the RMSE = 1.60 and MAE = 0.94.

## Improvements

An alternative approach would have been to create separate time series for each station, aggregate each series by the hour, and add 0 to each missing data point for the bike\_demand variable. After the time series have been created, various forecasting methods could have been applied to each series. However there are downsides to this approach:

* Multiple models need to be trained and tuned.
* The usage of neural networks would be required (LSTM/CNN).
* The models would not benefit from data from other stations.
* It would become more time consuming as the number of stations grow larger.

The current model could further be improved by creating a custom error measure that penalizes underestimation more heavily, since the purpose of demand planning is to have enough supply at any time for the existing demand. And because bicycles are non perishable goods, in theory, underestimating the demand is worse than overestimating it.

Additionally, using two models instead of one might produce better result. In these models, instead of estimating the overall demand, the number of Subscribers and Customers would be estimated separately. Feature selection and model tuning would need to be applied to each model, and since Customers type users are more affected by the weather, using hourly weather data could make the model more accurate. While using correct data instead of imputing it would produce a more consistent result.

Furthermore, a more complex model could also be applied, e.g. xgboost, which being a boosting algorithm, would most likely achieve a better result than the current random forest.

Finally, a newly proposed approach could also work better, which relies on a Graph Convolutional Neural Network and considers the underlying spatial or temporal correlations between stations to improve the prediction performance. https://arxiv.org/ftp/arxiv/papers/1712/1712.04997.pdf

## Conclusions

Overall the model performance (RMSE = 1.60, MAE = 0.94) shows, that the randomForest does a decent job estimating the normal demand. However it does not deal well with irregular demand (outliers). In order to improve the model performance, since removing the outliers are not an option, these must be either explained with additional data, or a more complex model is required (or both).

## Appendix

1. weather\_data\_imp.csv: Imputed weather data.
2. model\_tune\_rf.rds: tunRf model for mtry estimation.
3. model\_rf41.rds: randomForest model with all the features included.
4. model\_rf.rds: randomForest model with the best performing subset of features.
5. model\_final\_rf.rds: randomForest model with the best performing subset of features and optimal mtry.