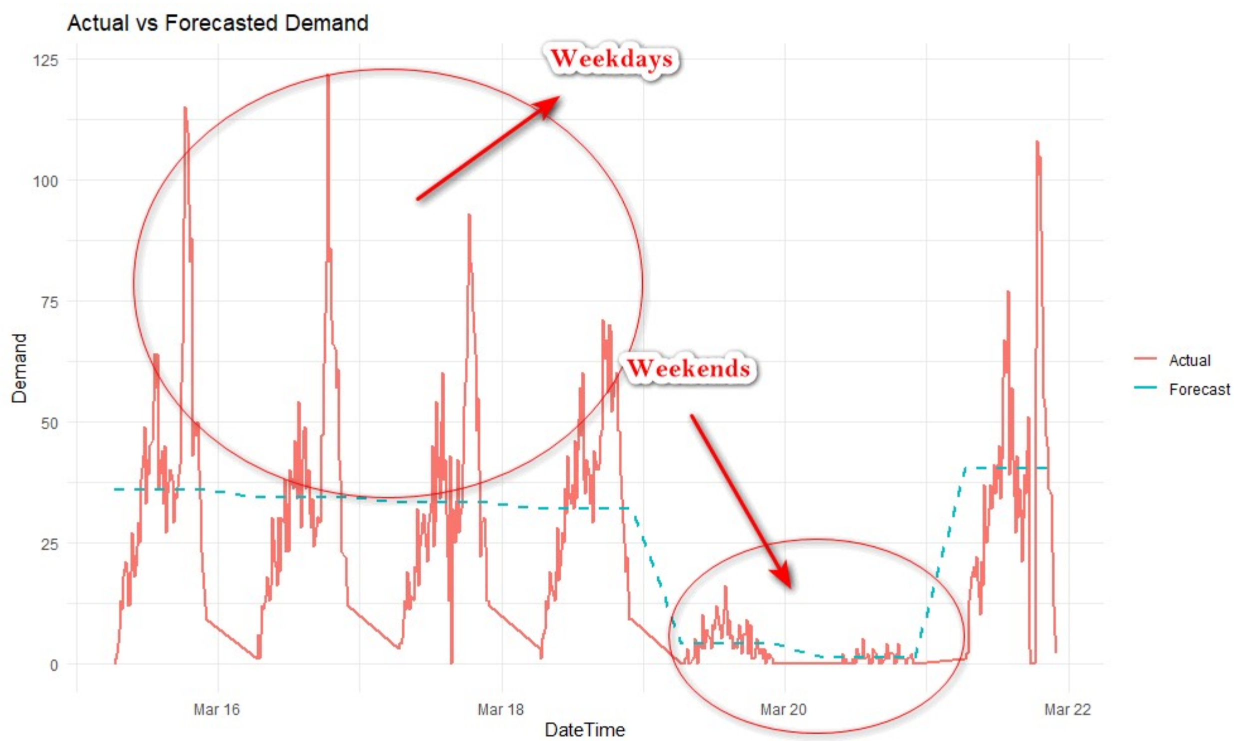
Final Project

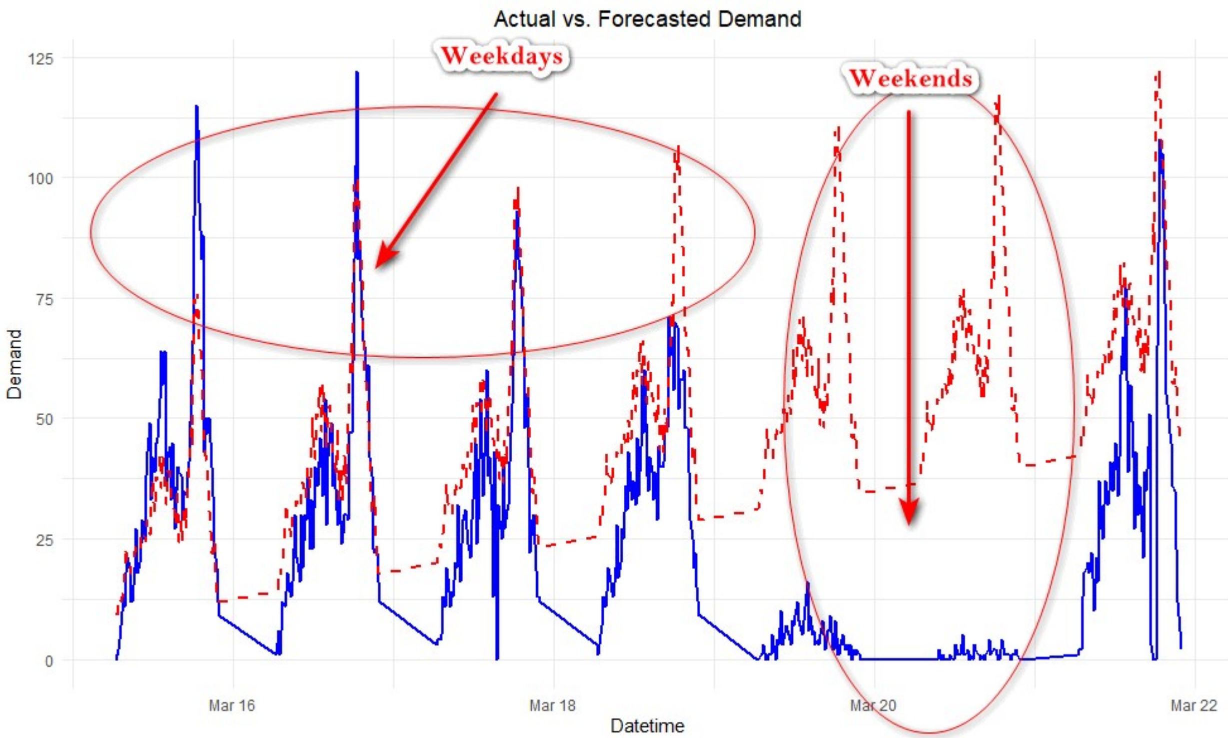
I chose a combination of the Arima and Naive methods for my final model. I am not skilled enough to combine the results in R so I will post separate visuals of each model. I combined the models because each model has a strength and weakness when compiling this data. The Naive method's strenght is to capture the weekly seasonality by day and by 15-minute interval. It worked well on weekends where the variability of usage was low but failed at capturing the high variability of usage on weekdays.

Naive Forecast



The Arima model did an excellent job of capturing the weekday variability of usage because it seeks to establish a trend over time with seasonality. It projected that trend/seasonality to weekends where the model did a poor job of capturing the low transit usage. My suggestion would be to combine the models for Arima ~ workdays/weekdays and Naive ~ weekends/slow holidays.

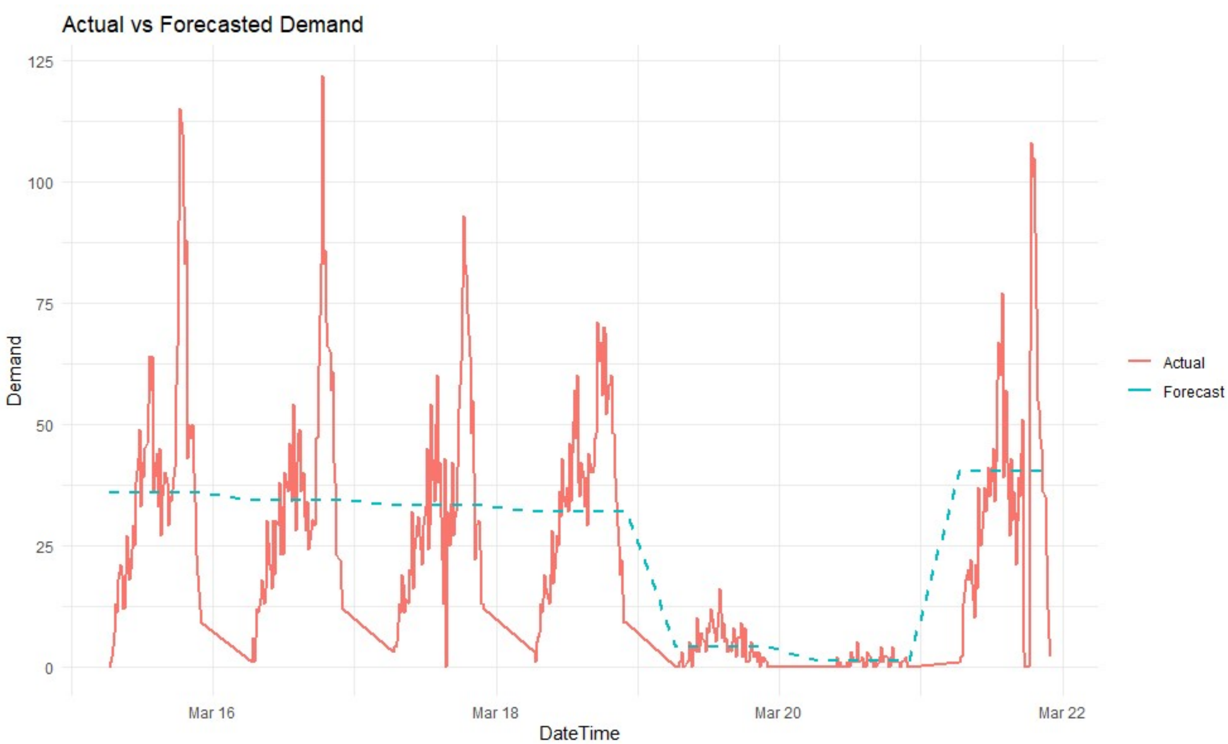
Arima Forecast



Technical Summary:

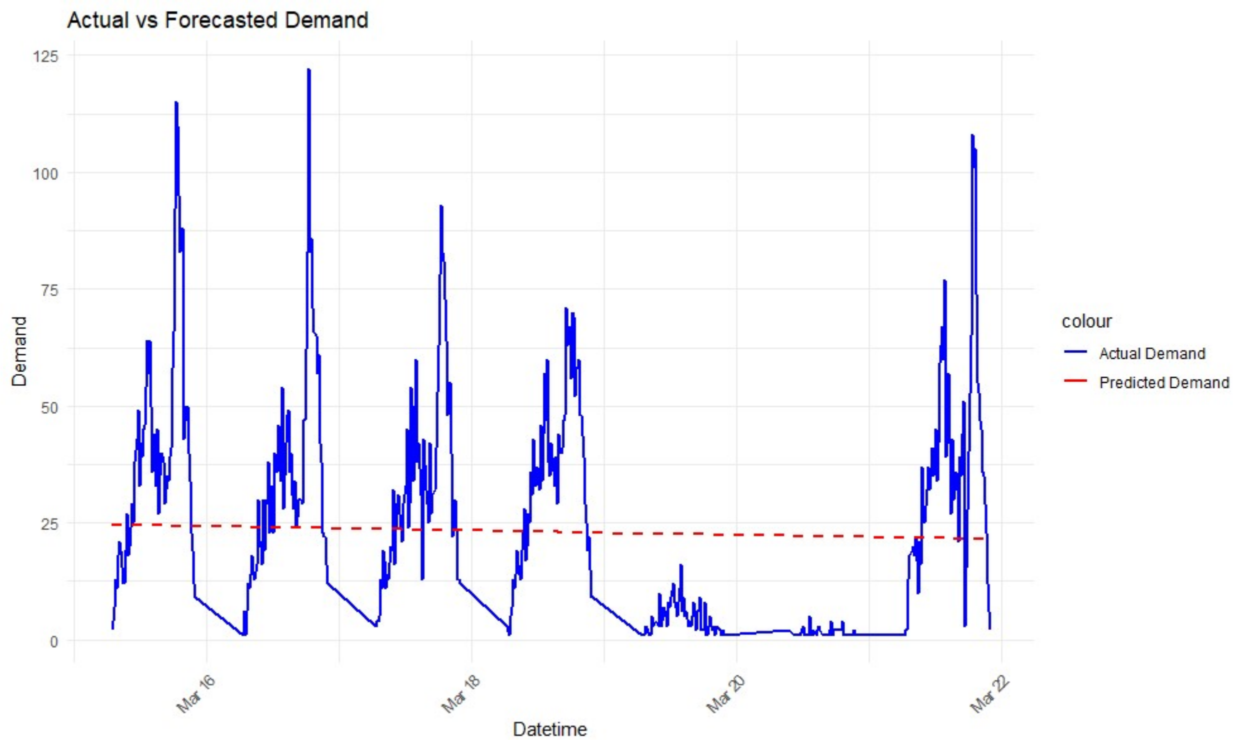
The Naive forecast as a benchmark.

- **Mean Absolute Error (MAE):** The average absolute error between the forecasted demand and the actual demand is approximately 12.66. This value indicates the average magnitude of errors in the forecasts, regardless of direction.
- **Root Mean Squared Error (RMSE):** The square root of the average squared differences between the forecasted demand and the actual demand is approximately 18.99. RMSE gives more weight to larger errors and is useful for understanding the magnitude of error.
- **Filtered Mean Absolute Percentage Error (MAPE):** After filtering out instances where the actual demand is zero to avoid division by zero issues, the MAPE is approximately 127.59%. This high value suggests that the forecast errors are large relative to the actual demand values, indicating significant discrepancies in the forecast accuracy.



Validation data of the Naive Forecast.

Linear Regression Model:



Comparing the performance metrics of the naive forecast and the linear regression model reveals interesting insights:

- **Naive Forecast Metrics:**
  - MAE: 12.6568765072166
  - RMSE: 18.9888449798315
  - Filtered MAPE: 127.594538853236
- **Linear Regression Model Metrics (Filtered for Demand > 0):**
  - MAE: 18.6840522177133
  - RMSE: 24.1018505149405
  - MAPE: 314.343608815863

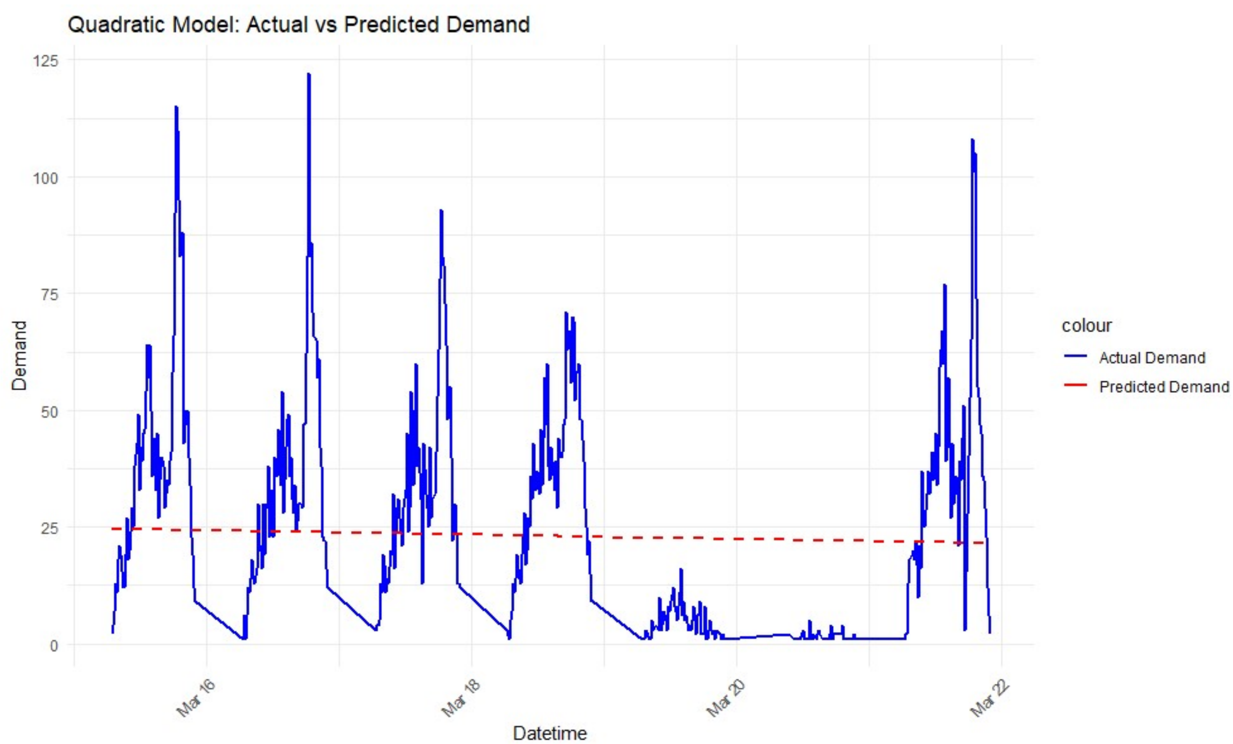
The naive forecast demonstrates lower MAE and RMSE values compared to the linear regression model, suggesting that it was more accurate in this specific case. The naive method is simpler and uses less information (only the previous period's demand) to make forecasts, which might have been sufficiently effective given the dataset's characteristics.

The MAPE values, especially, highlight a significant difference in percentage error between the two methods. The linear regression model's filtered MAPE is substantially higher, indicating that, on average, the linear model's predictions deviate more from the actual values in relative terms than the naive forecast.

Given these results, the naive forecast may be considered more reliable for this specific forecasting task.

The main takeaway is that the linear model does a poor job of capturing the weekend data in comparison to the weekday data. It attempts to model something in the middle without capturing any of the trend or seasonality in the data.

Quadratic model:



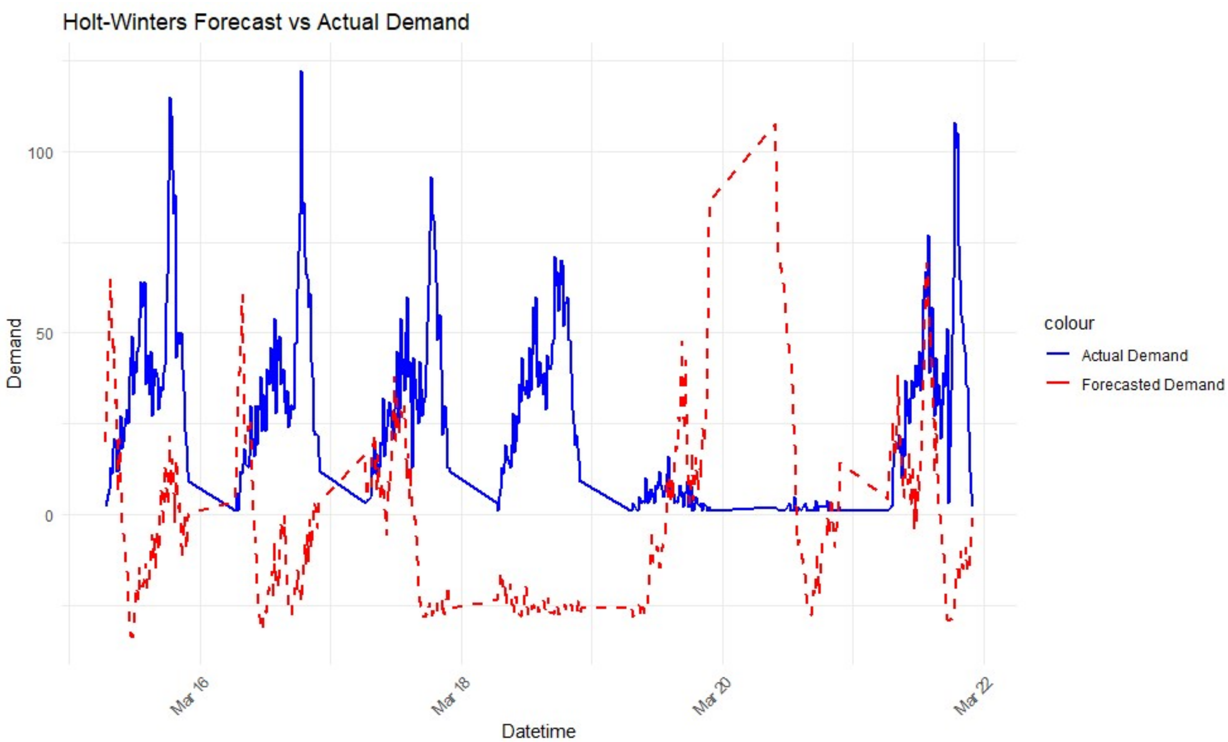
Summary of Error Metrics

- **Naive Forecast**:
  - MAE: 12.66
  - MAPE: 127.59
  - RMSE: 18.99
- **Linear Model**:
  - MAE: 18.68
  - MAPE: 314.34
  - RMSE: 24.10
- **Quadratic Model**:
  - MAE: 18.68
  - MAPE: 314.34
  - RMSE: 24.10

The naive forecast outperforms both the linear and quadratic models in terms of MAE and RMSE. This suggests that seasonally adjusted data of the demand in the Naive model is more accurate than using linear or quadratic regression models to predict future demand for this specific dataset.

The error metrics for the linear and quadratic models are identical, indicating that the quadratic term did not provide additional predictive power. This similarity in performance can be attributed to the quadratic term being undefined (NA) due to singularities. Essentially, the model couldn't find a significant quadratic relationship in the data, leading to the quadratic model collapsing back into a linear model.

Holt Winters Method:



Holt-Winters Model

- **MAE:** 39.73
- **MAPE:** 394.02%
- **RMSE:** 49.05

Seasonal Naive Forecast

- **MAE:** 12.66
- **MAPE:** 127.59%
- **RMSE:** 18.99

Linear Model

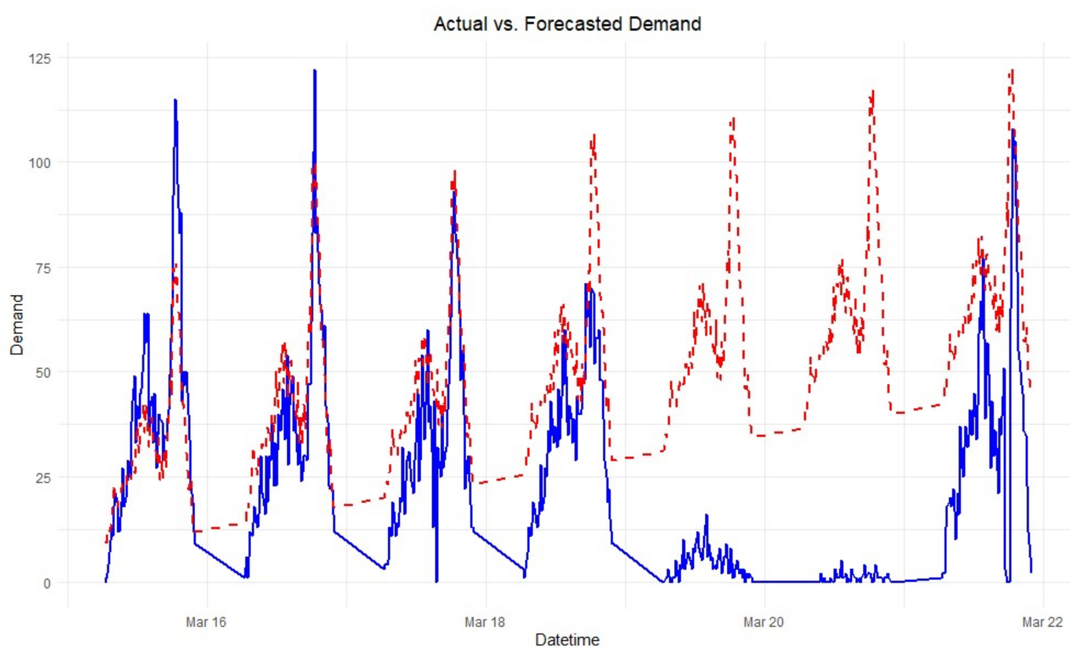
- **MAE:** 18.68
- **MAPE:** 314.34%
- **RMSE:** 24.10

Quadratic Model
(Since the Quadratic model performed similarly to the Linear model due to the singularity issue, its metrics are akin to the Linear model's metrics.)

The Holt-Winters method did not outperform the Seasonal Naive Forecast. Since the data is not perfectly seasonal within a 7 day period and the usage is significantly greater during the week and less during the weekend, the forecast does a poor job in capturing the seasonality. I tried a few different frequencies and still had poor results vs the naive model.

Arima

- **MAE:** 28.92261
- **Adjusted MAPE:** 824.6387%
- **RMSE:** 37.73276
- **MASE:** 4.376186



The Arima model does an excellent job of forecasting the weekdays and not the weekends. The Naive model does an excellent job of forecasting the weekends and not the weekdays. I spent a couple of hours trying to combine the weekdays Arima model with the weekend naive model and was unable to get the code to work. The reason for this is the 'seasonality' of the extreme usage (variability) on weekdays or workdays and the low variability of the weekend days which the naive model captured well due to its ability to forecast recent data well.

Due to the obvious strengths and weaknesses of opposing models, the error metrics do not represent the final model well.

Appendix

Data Cleaning:

```
# Check for missing values in bicuphi
> summary(bicuphi)
      X                  X.1                X.2
 Length:1325        Length:1325        Length:1325
 Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character
> # More detailed check
> sapply(bicuphi, function(x) sum(is.na(x)))
  X X.1 X.2
  0   0   0

~ Notes: There are no NA values in the data.  That does not mean that there could
be other types of data errors.


# Convert X.2 to numeric, coercing any non-numeric strings to NA
> bicuphi$X.2 <- as.numeric(as.character(bicuphi$X.2))
Warning message:
NAs introduced by coercion


# Check for newly created NAs in X.2 (if any non-numeric values were present)
> sum(is.na(bicuphi$X.2))
[1] 2


# Identify which rows have NAs in X.2 introduced by coercion
> na_rows <- which(is.na(bicuphi$X.2))
> print(na_rows)
[1] 1 2


# Optional: View the rows with NAs in X.2 to understand what the non-numeric
values were
> print(bicuphi[na_rows, ])
     X  X.1 X.2
1             NA
2 DATE TIME   NA


# Correctly count empty strings in each column
> empty_string_counts <- sapply(bicuphi, function(x) sum(x == ""))
> print(empty_string_counts)
  X X.1 X.2
  1   1  NA


    X  X.1 X.2
1             NA
2 DATE TIME   NA
> # Correcting the command to identify rows with empty strings
> empty_string_rows <- sapply(bicuphi, function(x) sum(x == ""))
> total_empty_strings <- sum(empty_string_rows)
> print(total_empty_strings)
[1] NA
```

```
>
> # Remove rows where X.2 is NA
> bicuphi_clean <- bicuphi[!is.na(bicuphi$X.2), ]
>
> # Check the dimensions to ensure rows are removed
> dim(bicuphi_clean)
[1] 1323    3
```

~ Notes: Two rows have been removed 1, 2.

```
# Renaming columns in the bicuphi_clean dataframe
> names(bicuphi_clean) <- c('date', 'time', 'demand')
>
> # Verify the changes
> head(bicuphi_clean)
      date time demand
3 1-Mar-05 6:30      1
4 1-Mar-05 6:45      2
5 1-Mar-05 7:00      4
6 1-Mar-05 7:15      0
7 1-Mar-05 7:30     10
8 1-Mar-05 7:45     13
```

~ Notes: Descriptive aspects of the columns have been resolved. Numeric type for demand is confirmed as numeric.

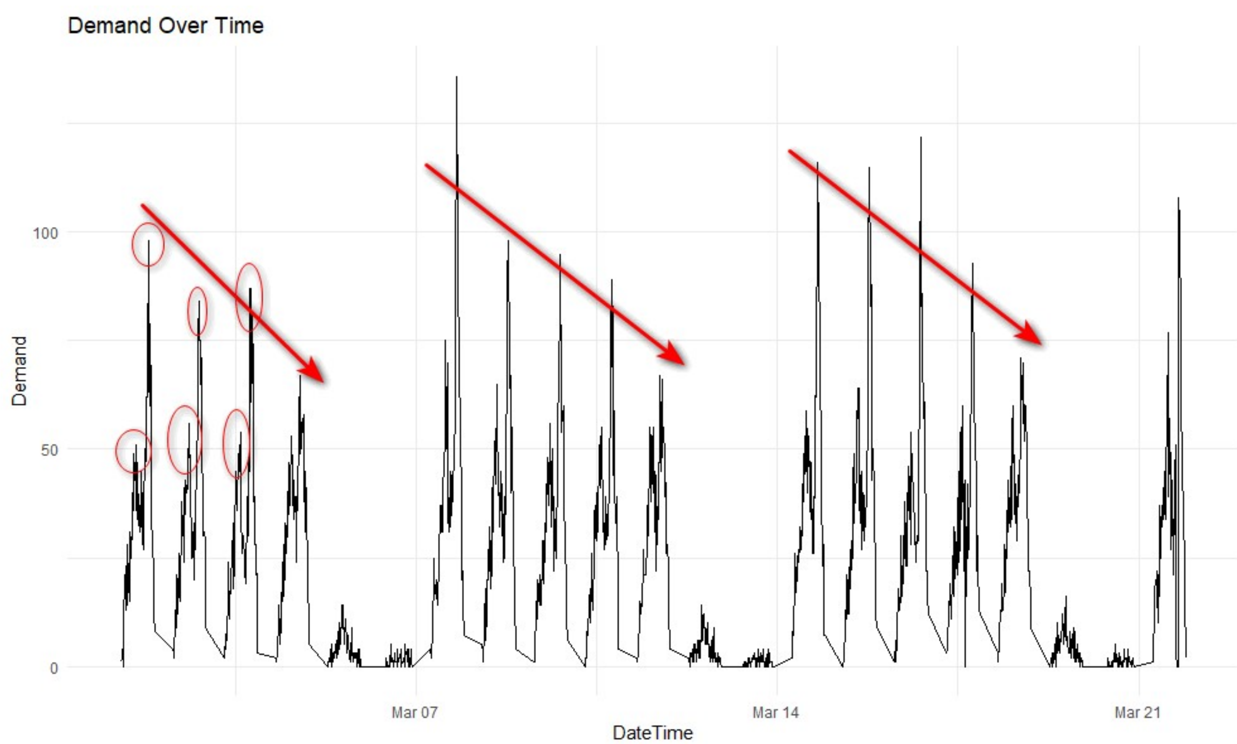
```
# Convert the 'date' column to Date format > bicuphi_clean$date <-
as.Date(bicuphi_clean$date, format="%d-%b-%y") > > # Assuming the 'time' column
is already in a suitable format, you might still want to combine 'date' and
'time' > # First, ensure 'time' is a character for concatenation >
bicuphi_clean$time <- as.character(bicuphi_clean$time) > > # Combine 'date' and
'time' into a single 'datetime' column > bicuphi_clean$datetime <-
as.POSIXct(paste(bicuphi_clean$date, bicuphi_clean$time), format="%Y-%m-%d
%H:%M") > > # Check if 'datetime' is correctly formatted and if time intervals
are consistent > summary(bicuphi_clean$datetime) Min. 1st Qu. Median Mean "2005-
03-01 06:30:00" "2005-03-06 10:22:30" "2005-03-11 14:15:00" "2005-03-11 14:15:00"
3rd Qu. Max. "2005-03-16 18:07:30" "2005-03-21 22:00:00"
```

~ Notes: Create a single datetime column for clarity and simplicity.

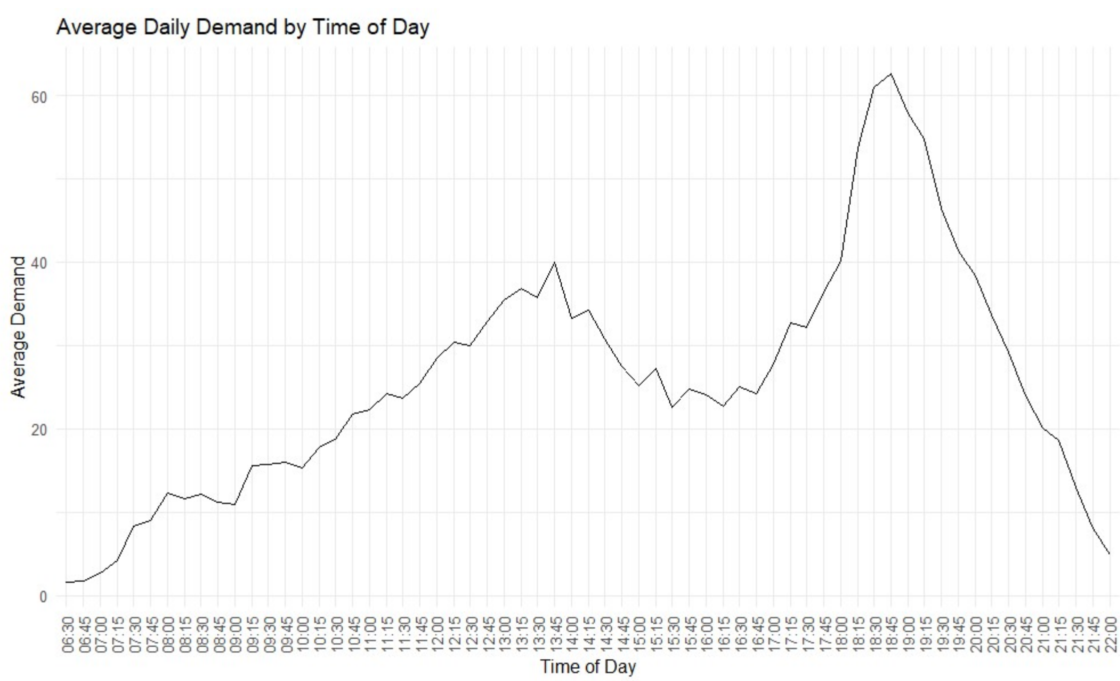
```
# Calculate differences between consecutive datetime entries
> time_diffs <- diff(bicuphi_clean$datetime)
>
> # Check if all differences are 15 minutes (900 seconds)
> all(time_diffs == 900) # Returns TRUE if all intervals are exactly 15 minutes
[1] FALSE
```

~ Notes: Not all intervals are 15 minutes

Data Visualization:



This is a time plot of the demand with a 'datetime' stamp on the x axis. Trend: Max usage is typically on Monday and decreases thru Sunday. Usage on weekdays builds steadily in the morning rush hour and peaks around the 5 pm rush hour.

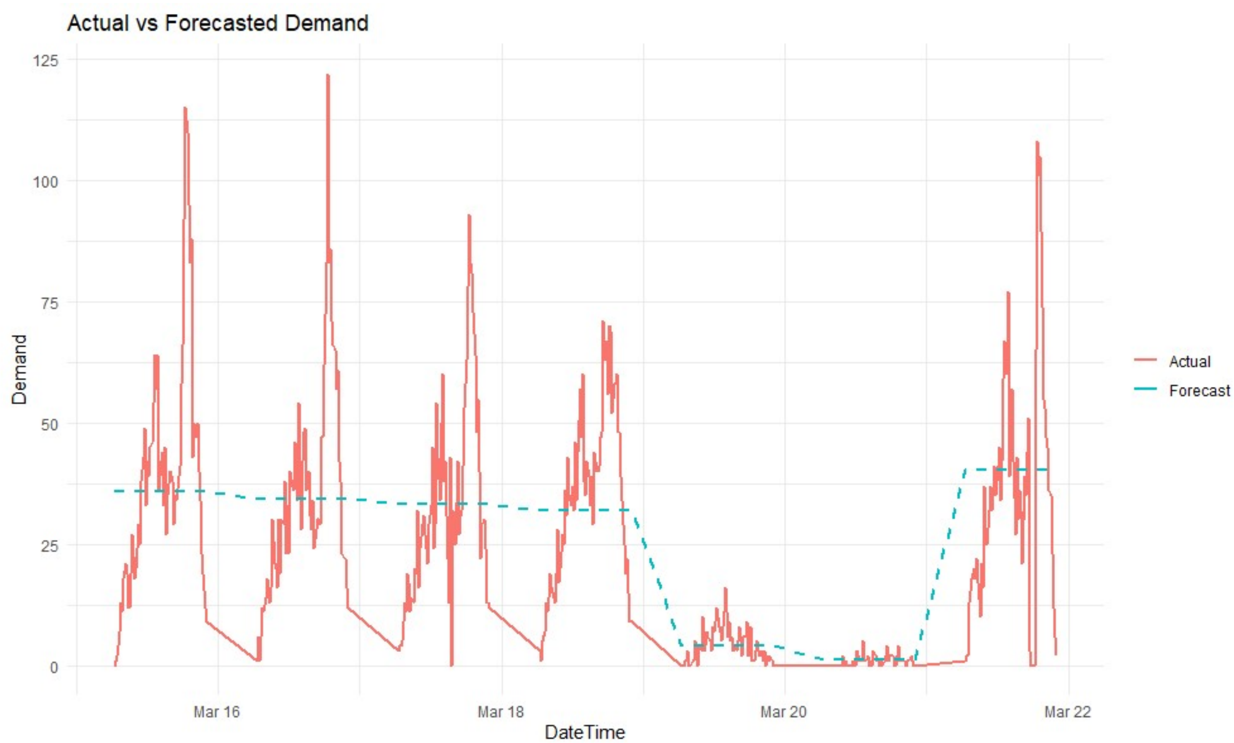


Average daily usage shows 2 peak periods that revolve around the workday hours. There is a spread our period in the morning and more concentrated usage period around evening rush hour.

Naive Forecast

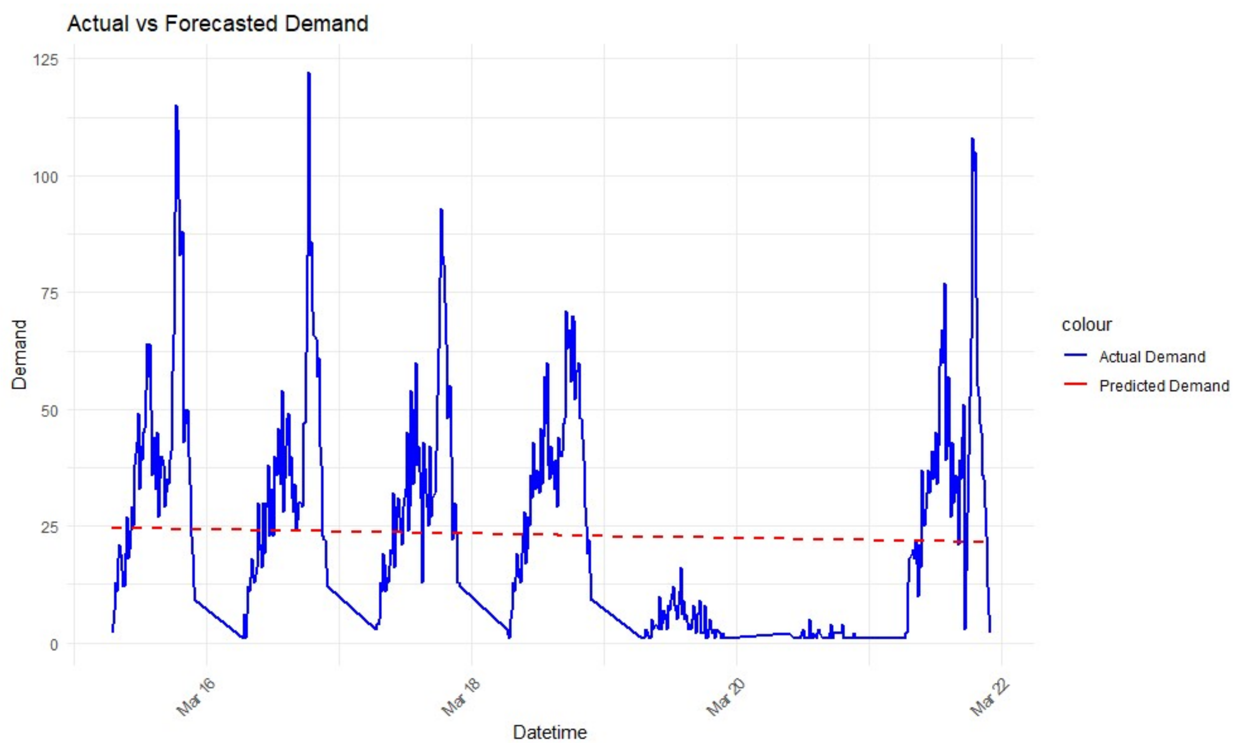
```
# Splitting the data
> training_set <- bicuphi_clean %>% filter(datetime < as.POSIXct("2005-03-15"))
> validation_set <- bicuphi_clean %>% filter(datetime >= as.POSIXct("2005-03-
15"))
>
> # Calculating weekly patterns from the training data
> weekly_pattern <- training_set %>%
+    mutate(day_of_week = wday(datetime, label = TRUE)) %>%
+    group_by(day_of_week) %>%
+    summarise(average_demand = mean(demand))
>
> # Viewing the weekly pattern
> print(weekly_pattern)
# A tibble: 7 × 2
  day_of_week average_demand
  <ord>                <dbl>
1 Sun                   1.12
2 Mon                  40.3
3 Tue                  35.9
4 Wed                  34.2
5 Thu                  33.2
6 Fri                  31.9
7 Sat                   4.11
>
> # Forecasting for the validation period
> validation_forecasts <- validation_set %>%
+    mutate(day_of_week = wday(datetime, label = TRUE)) %>%
+    left_join(weekly_pattern, by = "day_of_week") %>%
+    select(datetime, forecast_demand = average_demand)
>
> # Checking the forecasts
> head(validation_forecasts)
             datetime forecast_demand
1 2005-03-15 06:30:00         35.9127
2 2005-03-15 06:45:00         35.9127
3 2005-03-15 07:00:00         35.9127
4 2005-03-15 07:15:00         35.9127
5 2005-03-15 07:30:00         35.9127
6 2005-03-15 07:45:00         35.9127
>
> # Joining forecasts with actual demand
> validation_comparison <- validation_set %>%
+    select(datetime, actual_demand = demand) %>%
+    left_join(validation_forecasts, by = "datetime")
>
> # Calculating MAE
> mae <- mean(abs(validation_comparison$actual_demand -
validation_comparison$forecast_demand), na.rm = TRUE)
> print(paste("MAE:", mae))
[1] "MAE: 12.6568765072166"
>
> # Calculating RMSE
> rmse <- sqrt(mean((validation_comparison$actual_demand -
```

```
validation_comparison$forecast_demand)^2, na.rm = TRUE))
> print(paste("RMSE:", rmse))
[1] "RMSE: 18.9888449798315"
>
> # Ensure actual_demand and forecast_demand are numeric
> validation_comparison$actual_demand <-
as.numeric(validation_comparison$actual_demand)
> validation_comparison$forecast_demand <-
as.numeric(validation_comparison$forecast_demand)
>
> # Calculate MAPE
> mape <- mean(abs((validation_comparison$actual_demand -
validation_comparison$forecast_demand) / validation_comparison$actual_demand),
na.rm = TRUE) * 100
> print(paste("MAPE:", mape))
[1] "MAPE: Inf"

# Filtering out instances where actual_demand is zero to avoid division by zero
in MAPE calculation
> validation_comparison_filtered <- validation_comparison %>%
+    filter(actual_demand > 0)
>
> # Recalculating MAPE
> mape_filtered <- mean(abs((validation_comparison_filtered$actual_demand -
validation_comparison_filtered$forecast_demand) /
validation_comparison_filtered$actual_demand), na.rm = TRUE) * 100
> print(paste("Filtered MAPE:", mape_filtered))
[1] "Filtered MAPE: 127.594538853236"
>
```



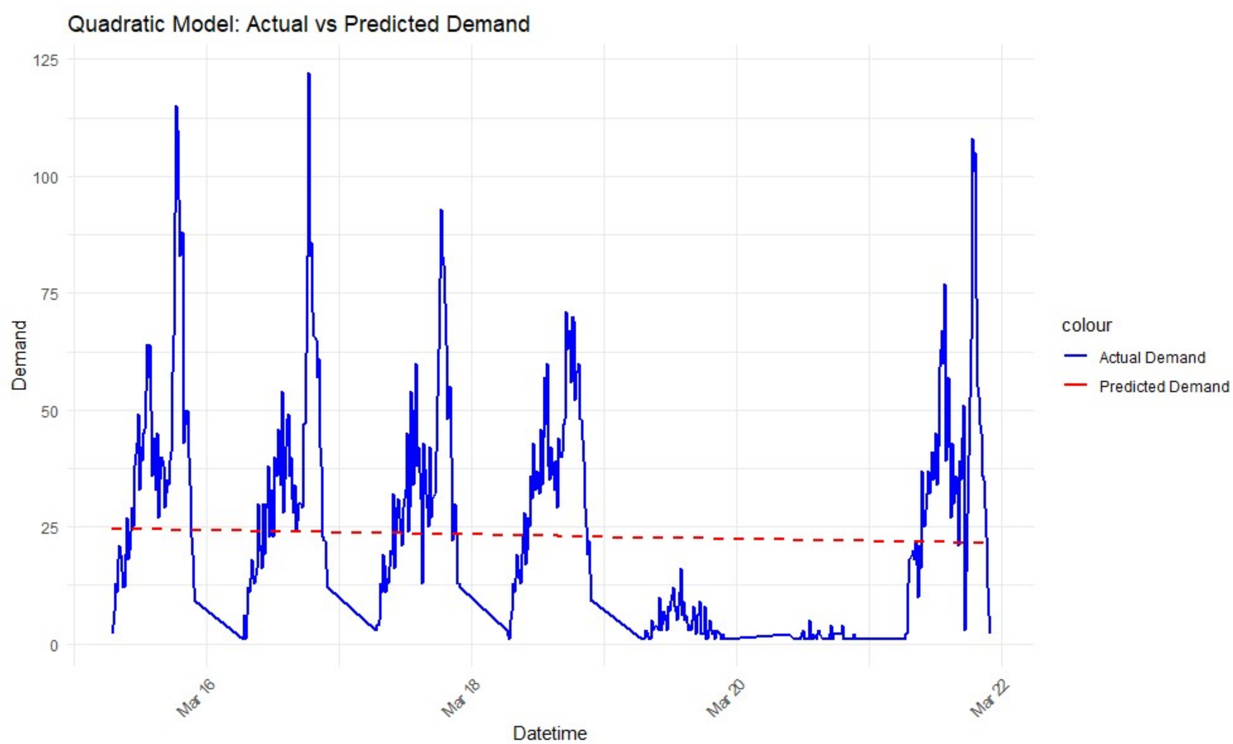
Linear Regression Model:

```
# Summary of the model
> summary(lm_model)

Call:
lm(formula = demand ~ datetime, data = training_set)

Residuals:
 Min 1Q Median 3Q Max
-29.989 -20.950 -1.847 13.339 107.994

Coefficients:
 Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.917e+03 2.548e+03 2.323 0.0204 *
datetime -5.305e-06 2.295e-06 -2.312 0.0210 *
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 22.9 on 814 degrees of freedom
Multiple R-squared: 0.006523,   Adjusted R-squared: 0.005302
F-statistic: 5.345 on 1 and 814 DF, p-value: 0.02104


# Print the error metrics
> print(paste("MAE:", mae))
[1] "MAE: 18.6840522177133"
> print(paste("MAPE:", mape))
[1] "MAPE: 314.343608815863"
> print(paste("RMSE:", rmse))
[1] "RMSE: 24.1018505149405"
```



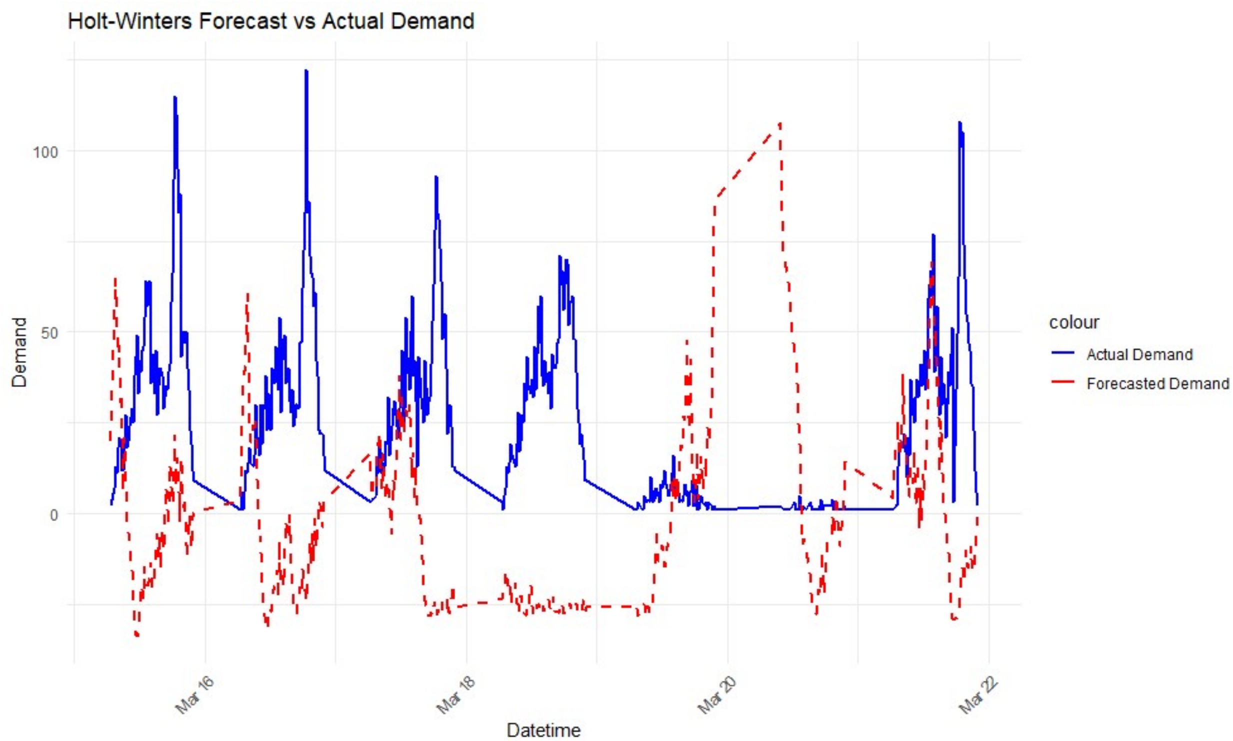
Quadratic Model:

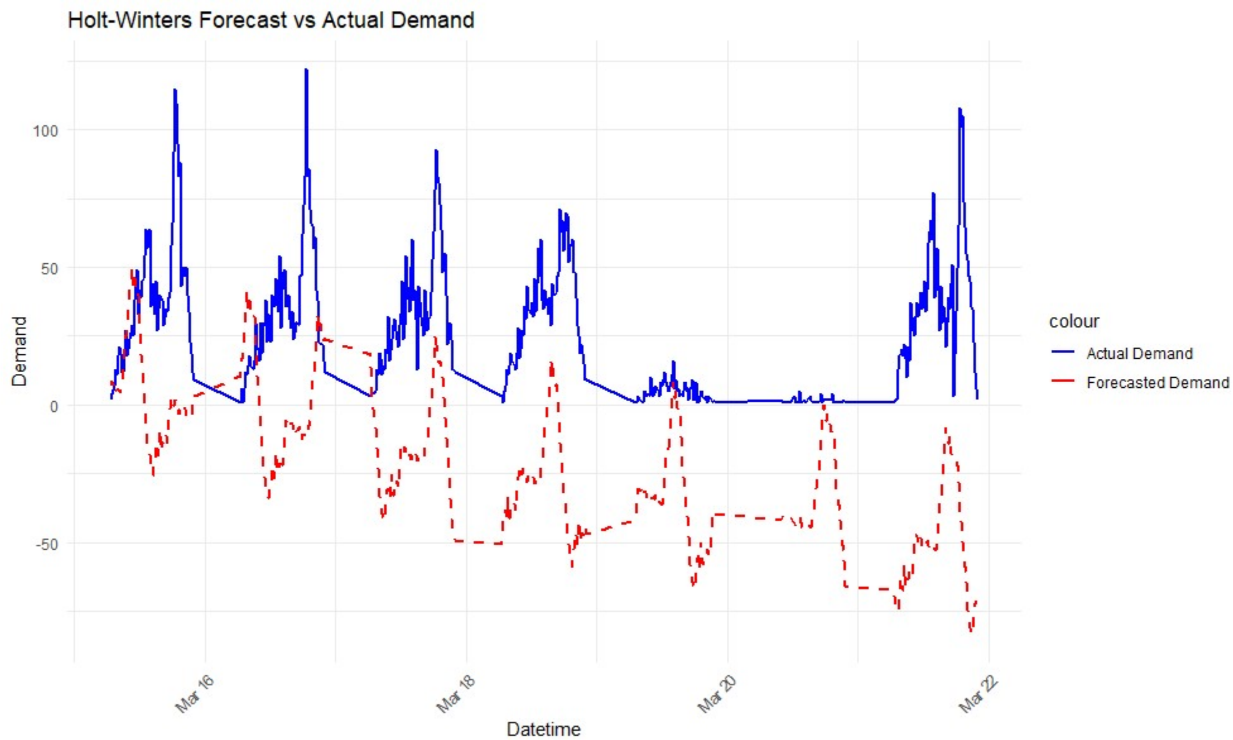
```
# Summary of the quadratic model
> summary(lm_quadratic_model)

Call:
lm(formula = demand ~ datetime_numeric + I(datetime_numeric^2),
    data = training_set_filtered)

Residuals:
    Min      1Q  Median      3Q     Max
-29.989 -20.950  -1.847  13.339 107.994

Coefficients: (1 not defined because of singularities)
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)            5.917e+03  2.548e+03   2.323   0.0204 *
datetime_numeric      -5.305e-06  2.295e-06  -2.312   0.0210 *
I(datetime_numeric^2)         NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 22.9 on 814 degrees of freedom
Multiple R-squared:  0.006523,	Adjusted R-squared:  0.005302
F-statistic: 5.345 on 1 and 814 DF,  p-value: 0.02104
```

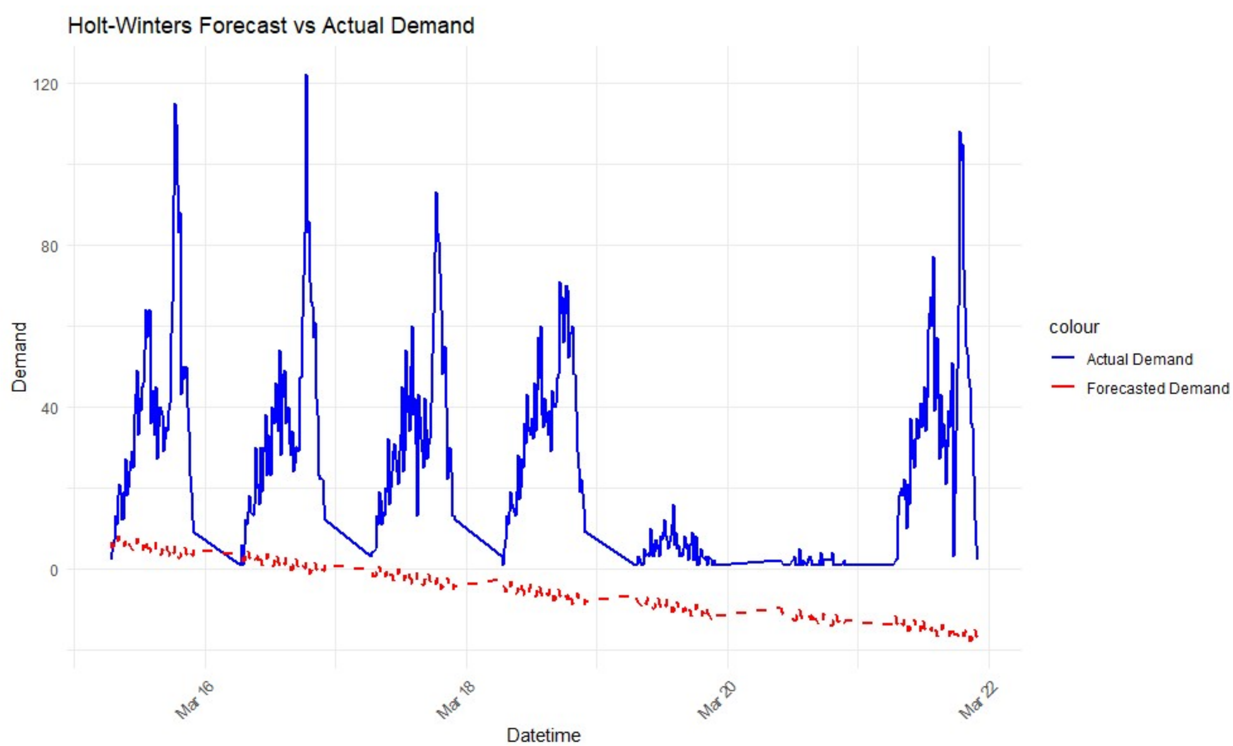


Holt Winters Method:

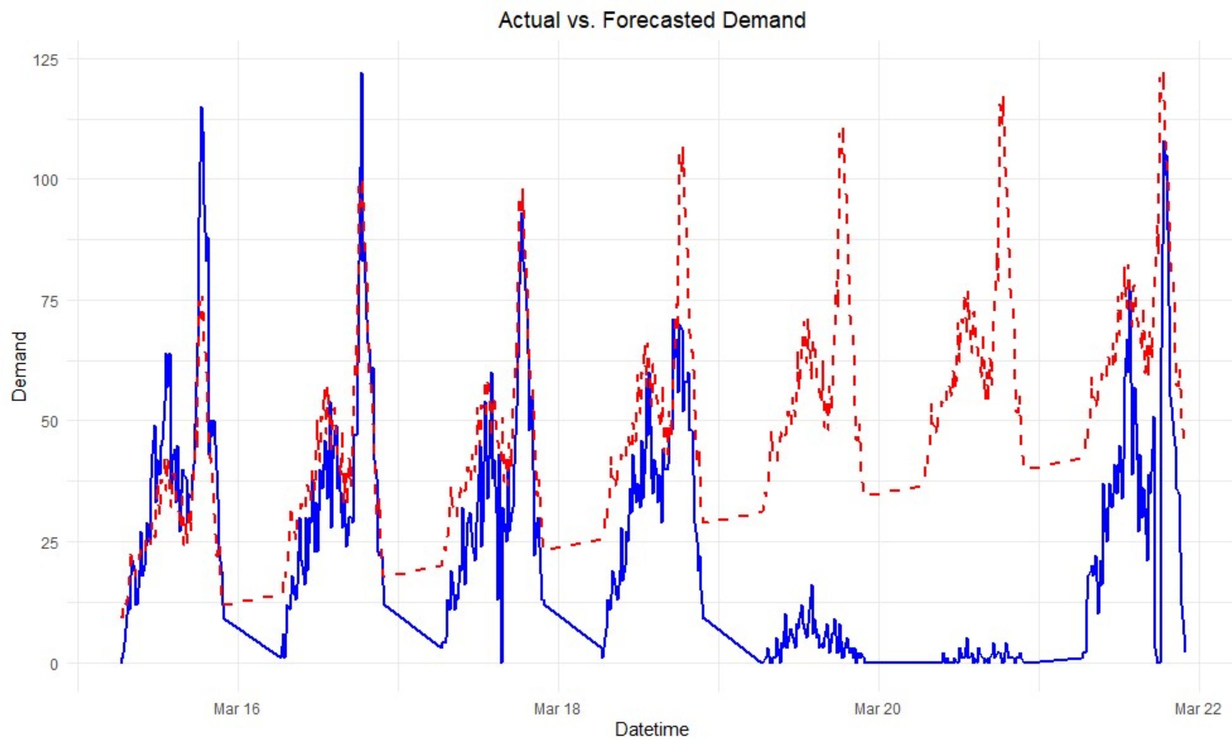
```
> # Print the error metrics
> print(paste("MAE:", mae))
[1] "MAE: 39.7316740676646"
> print(paste("MAPE:", mape))
[1] "MAPE: 394.020267682534"
> print(paste("RMSE:", rmse))
[1] "RMSE: 49.0510276675861"
>
```




Holt-Winters Forecast vs Actual Demand
100
50
0
-50
Demand
Mar 16
Mar 18
Mar 20
Mar 22
Datetime
colour
Actual Demand
Forecasted Demand


Holt-Winters Forecast vs Actual Demand
120
80
40
0
Demand
Mar 16
Mar 18
Mar 20
Mar 22
Datetime
colour
Actual Demand
Forecasted Demand

Arima



Actual vs. Forecasted Demand
Demand
125
100
75
50
25
0
Mar 16
Mar 18
Mar 20
Mar 22
Datetime