Linear Regression

Authors:

Andrew Sen
Atmin Sheth

Date:

9/25/2022

Overview of Linear Regression

Linear regression forms a model of data that takes predictor values to predict some quantitative target value in the data. Generally, linear regression creates some line of best fit for the data that can be used to predict values for new data. Despite the name, linear regression can be used for polynomial functions as well. Linear regression has low variance and is unlikely to overfit training data. However, linear regression tends to have high bias and may create a linear relationship that does not actually exist in the data, leading to underfit.

Data

This notebook will use a dataset found on the UCI Machine Learning Repository:

Fanaee-T, Hadi, and Gama, Joao, ‘Event labeling combining ensemble detectors and background knowledge’, Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg, https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset (https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset).

The data describes hourly bike rental numbers from the Capital Bikeshare system between 2011 and 2012.

The predictors include:

- instant: record index
- dteday: date
- season: season (1:winter, 2:spring, 3:summer, 4:fall)
- yr: year (0:2011, 1:2012)
- mnth: month (1 - 12)
- hr: hour (0-23)
- holiday: (0:not a holiday, 1:holiday)
- weekday: (0:Sunday - 6:Saturday)
- workingday: (0:not a workday, 1:not weekend and not holiday)
- weathersit: hourly weather
    - 1: clear, few clouds, partly cloudy
    - 2: mist+cloudy, mist+broken clouds, mist+few clouds, mist
    - 3: light snow, light rain+thunderstorm+scattered clouds, light rain+scattered clouds
    - 4: heavy rain+ice pallets+thunderstorm+mist, snow+fog
- temp: normalized temperature in Celsius
- atemp: normalized feeling temperature in Celsius
- hum: normalized humidity divided by 100
- windspeed: normalized wind speed divided by 67

The possible target columns include:

- casual: number of rentals from casual users
- registered: number of rentals from registered users
- cnt: total rentals

We will be predicting cnt.

Data Cleaning

First, we will read in the data. Then, we will clean the data by removing rows with NAs and removing the instant and dteday columns. We will also remove the casual and registered columns because they are not independent from cnt.

```
bikesharing <- read.csv("data/bike-sharing.csv")
bikesharing <- bikesharing[,c(3:14, 17)] #remove instant, dteday, casual, registered
bikesharing <- bikesharing[complete.cases(bikesharing),] #remove incomplete rows
```

Data Exploration

First, we will divide the data into training and test data with an 80/20 split.

```
set.seed(1234)
i <- sample(1:nrow(bikesharing), nrow(bikesharing)*0.8, replace=FALSE)
train <- bikesharing[i,]
test <- bikesharing[-i,]
```

Now, we will explore the training data. First, we will see summaries of all the columns.

```
summary(train)
```

```
##      season           yr              mnth              hr       
##  Min.   :1.000   Min.   :0.0000   Min.   : 1.000   Min.   : 0.00  
##  1st Qu.:2.000   1st Qu.:0.0000   1st Qu.: 4.000   1st Qu.: 6.00  
##  Median :3.000   Median :1.0000   Median : 7.000   Median :12.00  
##  Mean   :2.504   Mean   :0.5064   Mean   : 6.549   Mean   :11.55  
##  3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:10.000   3rd Qu.:18.00  
##  Max.   :4.000   Max.   :1.0000   Max.   :12.000   Max.   :23.00  
##     holiday           weekday        workingday       weathersit   
##  Min.   :0.00000   Min.   :0.000   Min.   :0.0000   Min.   :1.000  
##  1st Qu.:0.00000   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:1.000  
##  Median :0.00000   Median :3.000   Median :1.0000   Median :1.000  
##  Mean   :0.02877   Mean   :3.011   Mean   :0.6821   Mean   :1.421  
##  3rd Qu.:0.00000   3rd Qu.:5.000   3rd Qu.:1.0000   3rd Qu.:2.000  
##  Max.   :1.00000   Max.   :6.000   Max.   :1.0000   Max.   :4.000  
##       temp            atemp             hum           windspeed     
##  Min.   :0.0200   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:0.3400   1st Qu.:0.3333   1st Qu.:0.4800   1st Qu.:0.1045  
##  Median :0.5000   Median :0.4848   Median :0.6300   Median :0.1940  
##  Mean   :0.4972   Mean   :0.4760   Mean   :0.6265   Mean   :0.1896  
##  3rd Qu.:0.6600   3rd Qu.:0.6212   3rd Qu.:0.7800   3rd Qu.:0.2537  
##  Max.   :1.0000   Max.   :0.9848   Max.   :1.0000   Max.   :0.8507  
##       cnt       
##  Min.   :  1.0  
##  1st Qu.: 41.0  
##  Median :144.0  
##  Mean   :190.8  
##  3rd Qu.:282.0  
##  Max.   :977.0  
```

We can use dim() to find the number of rows and columns.

```
dim(train)
```

```
## [1] 13903    13
```

We can use head() to see the first few rows of the training data.

```
head(train)
```

	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp
	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<dbl>
7452	4	0	11	2	0	6	0	1	0.24
8016	4	0	12	15	0	1	1	2	0.46
7162	4	0	10	0	0	1	1	1	0.26
8086	4	0	12	13	0	4	1	1	0.30

	season <int>	yr <int>	mnth <int>	hr <int>	holiday <int>	weekday <int>	workingday <int>	weathersit <int>	temp <dbl>
9196	1	1	1	1	0	2	1	2	0.32
623	1	0	1	4	0	6	0	1	0.16

6 rows | 1-10 of 14 columns

Similarly, we can use tail() to see the last few rows.

```
tail(train)
```

	season <int>	yr <int>	mnth <int>	hr <int>	holiday <int>	weekday <int>	workingday <int>	weathersit <int>	temp <dbl>
7800	4	0	11	14	0	6	0	2	0.48
1371	1	0	3	9	0	3	1	1	0.28
10410	1	1	3	20	0	3	1	1	0.62
14797	3	1	9	17	0	4	1	1	0.72
4207	3	0	6	4	0	3	1	1	0.66
12111	2	1	5	19	0	4	1	1	0.70

6 rows | 1-10 of 14 columns

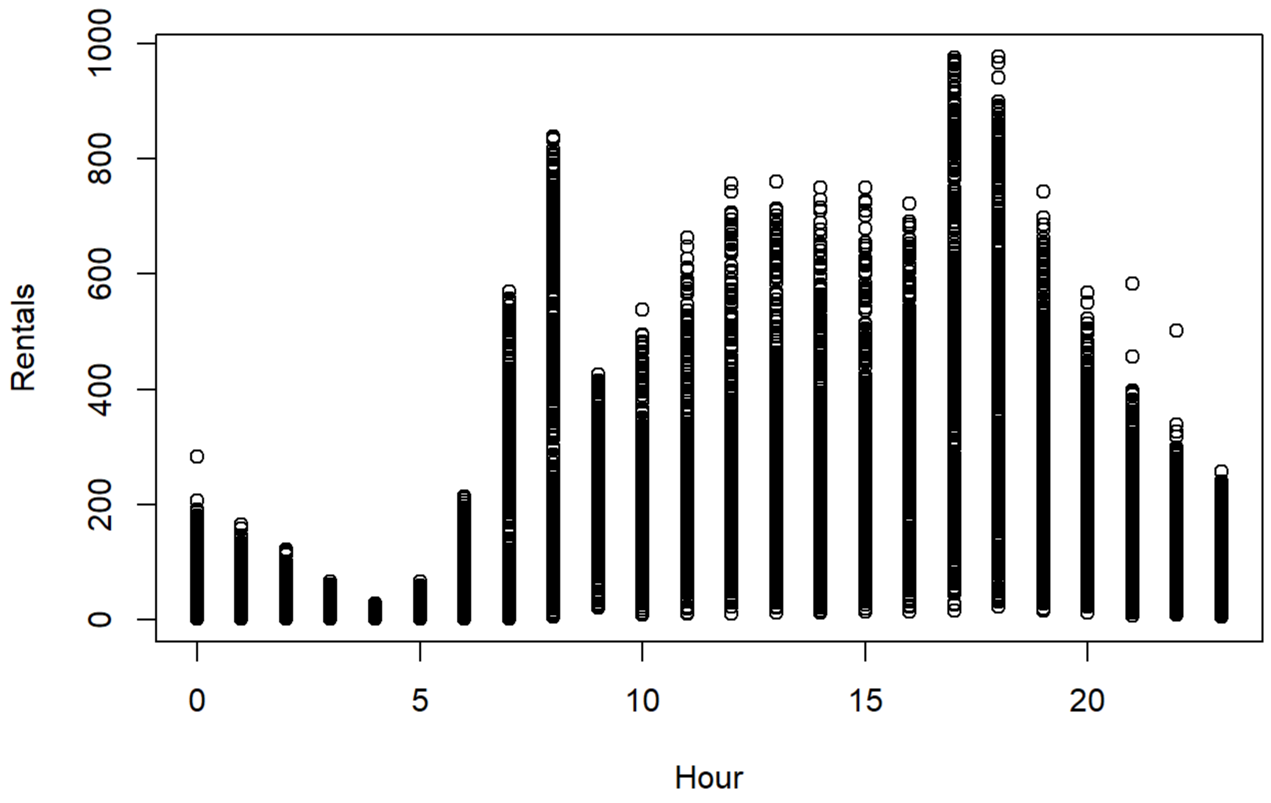
Finally, we can use str() to view the structure of the training data.

```
str(train)
```

```
## 'data.frame':    13903 obs. of  13 variables:
##  $ season    : int  4 4 4 4 1 1 4 2 1 2 ...
##  $ yr        : int  0 0 0 0 1 0 1 1 0 1 ...
##  $ mnth      : int  11 12 10 12 1 1 10 4 2 6 ...
##  $ hr        : int  2 15 0 13 1 4 5 16 12 20 ...
##  $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ weekday   : int  6 1 1 4 2 6 2 2 5 0 ...
##  $ workingday: int  0 1 1 1 1 0 1 1 1 0 ...
##  $ weathersit: int  1 2 1 1 2 1 2 1 1 1 ...
##  $ temp      : num  0.24 0.46 0.26 0.3 0.32 0.16 0.56 0.62 0.22 0.62 ...
##  $ atemp     : num  0.258 0.455 0.303 0.273 0.303 ...
##  $ hum       : num  0.65 0.72 0.87 0.49 0.93 0.69 0.83 0.21 0.47 0.57 ...
##  $ windspeed : num  0.0896 0.0896 0 0.3582 0.2537 ...
##  $ cnt       : int  46 148 23 124 9 3 43 485 71 302 ...
```

Next, we’ll graph the number of bike rentals over the hour of the day.

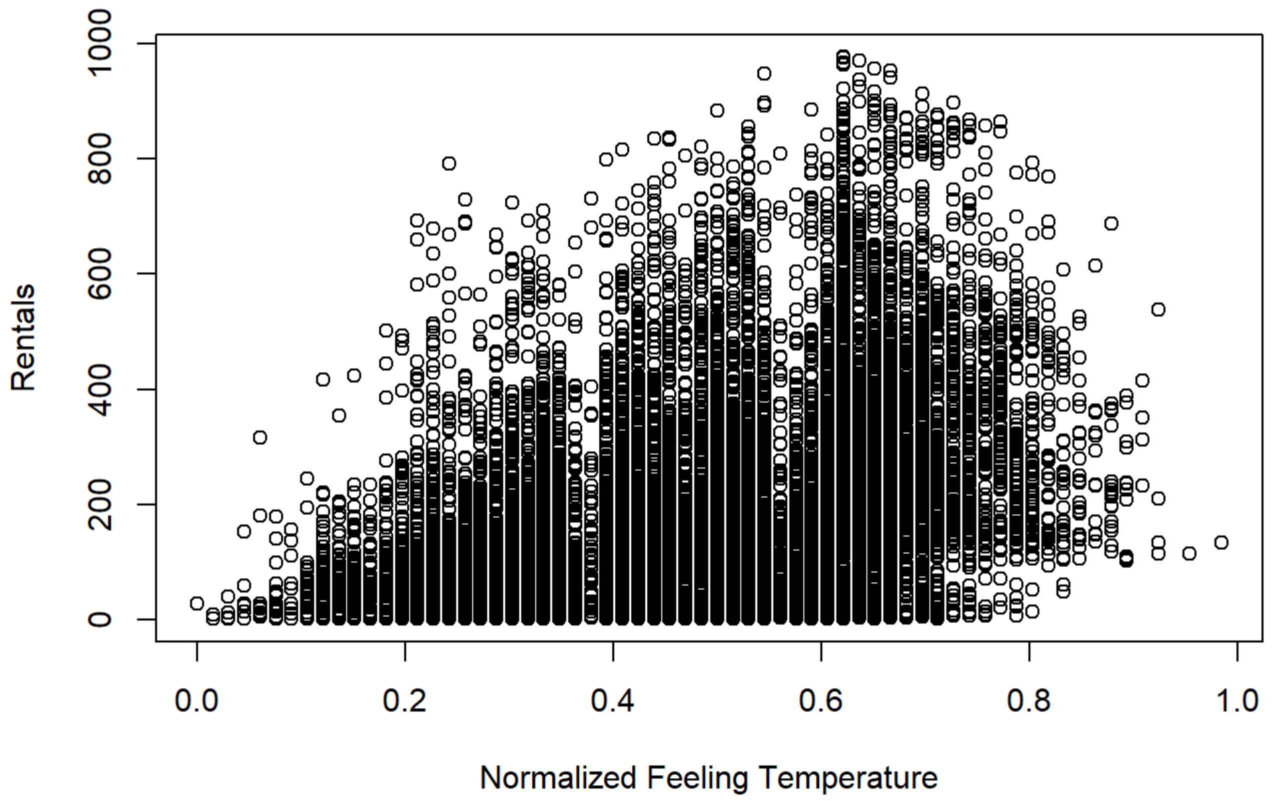
```
plot(train$hr, train$cnt, xlab="Hour", ylab="Rentals")
```



The graph shows that early morning and the afternoon tend to be the most popular times for renting a bike.

We will also plot bike rentals over feeling temperature.

```
plot(train$atemp, train$cnt, xlab="Normalized Feeling Temperature", ylab="Rentals")
```



This shows that bike rentals tend to increase if the temperature is warmer.

Simple Linear Regression

We will create a simple linear regression model using feeling temperature as our single predictor.

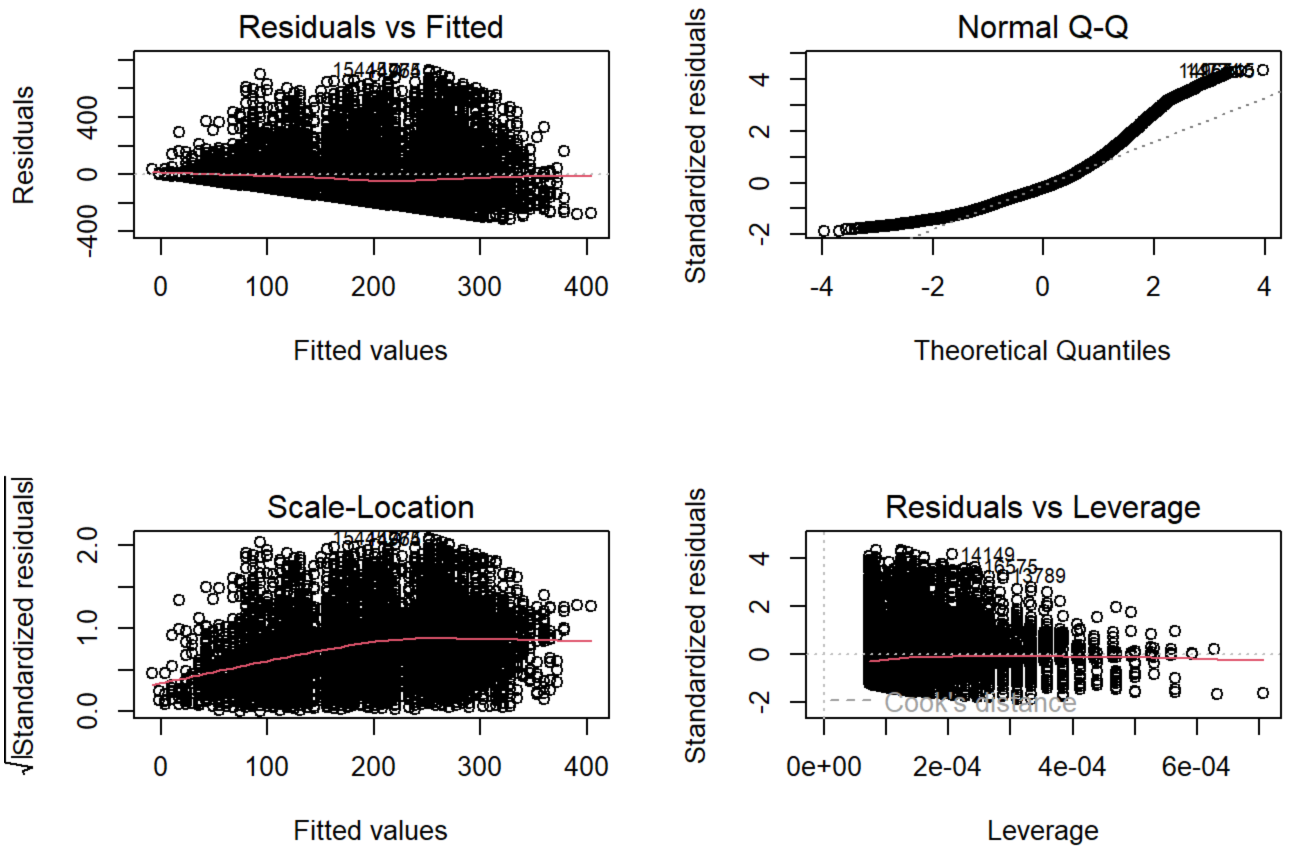
```
lm1 <- lm(cnt~atemp, data=train)
summary(lm1)
```

```
##
## Call:
## lm(formula = cnt ~ atemp, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -314.31 -112.29  -34.04   78.69  728.14
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -8.454      4.192  -2.017   0.0437 *
## atemp        418.538      8.286  50.512   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 167.4 on 13901 degrees of freedom
## Multiple R-squared:  0.1551, Adjusted R-squared:  0.155
## F-statistic:  2551 on 1 and 13901 DF,  p-value: < 2.2e-16
```

The model estimates that for every unit increase in feeling temperature, there will be 418.538 more bike rentals. R-squared is 0.1551, indicating that this linear model is a poor predictor of bike rentals. We would like to see an R-squared value that is closer to 1. The residuals are the differences between the predicted values for cnt and the actual values for cnt. The residual standard error (RSE) is the standard deviation of the residuals, and we would like it to be as low as possible. When we make other models later, we can compare them by comparing their RSE values.

We can also judge our model by looking at the residuals.

```
par(mfrow=c(2,2))
plot(lm1)
```



The Residuals vs. Fitted graph tells us if the relationship is linear. If the residuals are equally spread around a horizontal line without distinct patterns, this indicates the relationship isn't non-linear. The graph for this model does not seem evenly spread out, suggesting it is not linear.

Normal Q-Q shows if the residuals are normally distributed. If the residuals are plotted in a straight line, the relationship is likely linear. In our case, the plotted points seem to greatly deviate after a certain point. This indicates that the relationship is not linear.

Spread-Location shows if the residuals are spread equally along the range of predictors. Similar to the Residuals vs. Fitted graph, we want to see a straight, horizontal line with the residuals spread evenly around it. The line in our Scale-Location graph is not horizontal as it has a steep slope until around 200 on the x-axis. This further indicates the relationship is not linear.

Finally, Residuals vs. Leverage tells us if there were any influential cases. Note that even if a data point is an outlier, that does not necessarily mean the regression would have been different had it not been present. An influential point is an extreme case that changes the regression due to its presence. The red dotted lines mark the Cook's distance. Points beyond this line are influential points. Our model has not influential points.

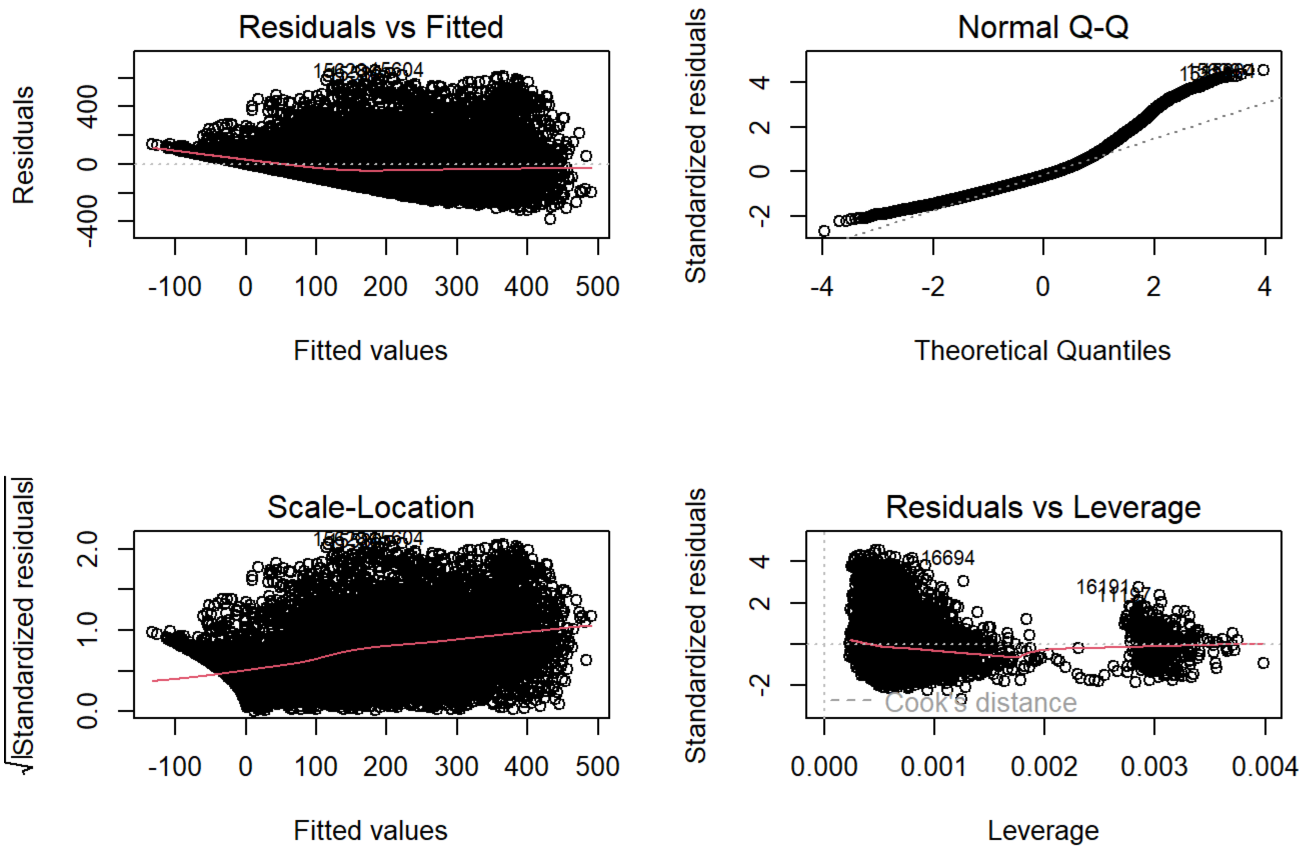
Multiple Linear Regression

Now we will attempt to make a better model by using multiple predictors. I will use every predictor except season because it corresponds to month, workingday because it corresponds to weekday and holiday, and temp because it correlates with atemp.

```
lm2 <- lm(cnt~hr+atemp+hum+windspeed+weathersit+weekday+holiday+yr+mnth, data=train)
summary(lm2)
```

```
##
## Call:
## lm(formula = cnt ~ hr + atemp + hum + windspeed + weathersit +
##     weekday + holiday + yr + mnth, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -385.05  -94.31  -28.13   61.33  652.78
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -14.4835     7.5845  -1.910 0.056204 .
## hr             7.6400     0.1866  40.937  < 2e-16 ***
## atemp        335.2978     7.3847  45.405  < 2e-16 ***
## hum         -198.7818     7.7990 -25.488  < 2e-16 ***
## windspeed     40.3278    10.6670   3.781 0.000157 ***
## weathersit    -3.9371     2.1625  -1.821 0.068683 .
## weekday        1.5715     0.6098   2.577 0.009972 **
## holiday      -26.7049     7.3211  -3.648 0.000266 ***
## yr            82.4723     2.4435  33.752  < 2e-16 ***
## mnth           5.2171     0.3693  14.128  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 143.4 on 13893 degrees of freedom
## Multiple R-squared:  0.3805, Adjusted R-squared:  0.3801
## F-statistic:   948 on 9 and 13893 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lm2)
```



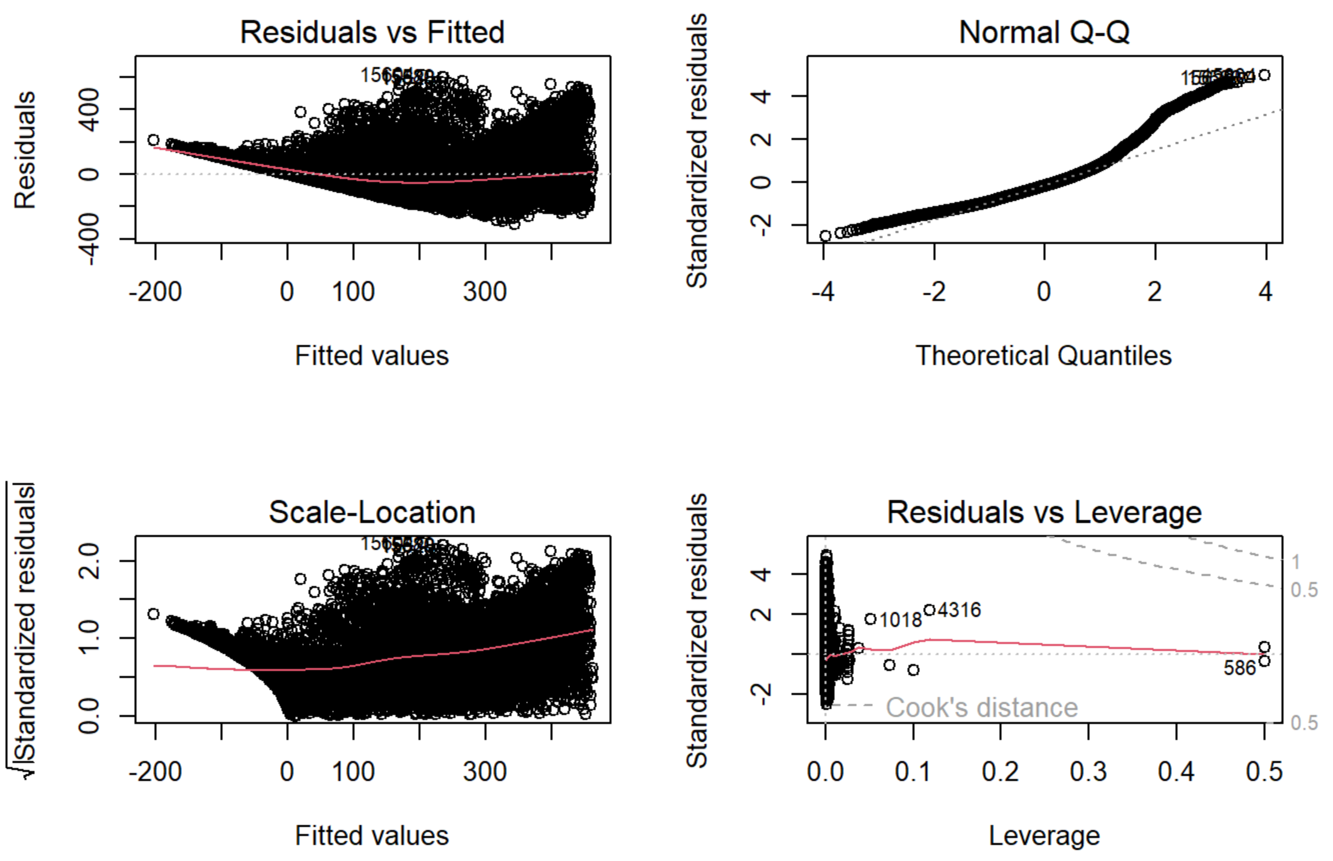
Polynomial Regression

Finally, we will create a polynomial regression model using the same predictors as our second linear regression model.

```
lm3 <- lm(cnt~poly(yr)+poly(mnth, degree=3)+poly(hr,degree=3)+poly(holiday)+poly(weekday, deg
ree=3)+poly(weathersit, degree=3)+poly(atemp, degree=3)+poly(hum, degree=3)+poly(windspeed, d
egree=3), data=train)
summary(lm3)
```

```
##
## Call:
## lm(formula = cnt ~ poly(yr) + poly(mnth, degree = 3) + poly(hr,
##     degree = 3) + poly(holiday) + poly(weekday, degree = 3) +
##     poly(weathersit, degree = 3) + poly(atemp, degree = 3) +
##     poly(hum, degree = 3) + poly(windspeed, degree = 3), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -309.15  -81.49  -20.25   53.64  611.63
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     190.761      1.048 182.070  < 2e-16 ***
## poly(yr)                       5231.497    125.198  41.786  < 2e-16 ***
## poly(mnth, degree = 3)1        2218.464    139.406  15.914  < 2e-16 ***
## poly(mnth, degree = 3)2       -2134.955    228.860  -9.329  < 2e-16 ***
## poly(mnth, degree = 3)3         -76.936    132.911  -0.579  0.56270
## poly(hr, degree = 3)1          7921.785    136.395  58.080  < 2e-16 ***
## poly(hr, degree = 3)2         -7937.703    131.939 -60.162  < 2e-16 ***
## poly(hr, degree = 3)3         -5103.417    136.033 -37.516  < 2e-16 ***
## poly(holiday)                  -506.394    125.437  -4.037 5.44e-05 ***
## poly(weekday, degree = 3)1      619.749    124.602   4.974 6.64e-07 ***
## poly(weekday, degree = 3)2     -373.317    124.246  -3.005  0.00266 **
## poly(weekday, degree = 3)3      102.199    124.586   0.820  0.41206
## poly(weathersit, degree = 3)1 -2043.186    145.711 -14.022  < 2e-16 ***
## poly(weathersit, degree = 3)2  -899.573    126.121  -7.133 1.03e-12 ***
## poly(weathersit, degree = 3)3    86.478    123.676   0.699  0.48442
## poly(atemp, degree = 3)1       3303.780    239.280  13.807  < 2e-16 ***
## poly(atemp, degree = 3)2       -602.595    136.462  -4.416 1.01e-05 ***
## poly(atemp, degree = 3)3      -1765.558    130.860 -13.492  < 2e-16 ***
## poly(hum, degree = 3)1         -991.274    168.755  -5.874 4.35e-09 ***
## poly(hum, degree = 3)2         -376.554    131.163  -2.871  0.00410 **
## poly(hum, degree = 3)3          154.818    125.453   1.234  0.21720
## poly(windspeed, degree = 3)1   -417.512    133.761  -3.121  0.00180 **
## poly(windspeed, degree = 3)2   -801.758    124.714  -6.429 1.33e-10 ***
## poly(windspeed, degree = 3)3    -95.497    124.279  -0.768  0.44226
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 123.5 on 13879 degrees of freedom
## Multiple R-squared:  0.5405, Adjusted R-squared:  0.5398
## F-statistic: 709.8 on 23 and 13879 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lm3)
```



The simple linear regression model appears to be the worst one based on the adjusted R-squared value. The residual plots make it apparent that the relationship displayed between cnt and atemp is not very linear, so it makes sense that the simple linear regression model would suffer from underfit. The multiple linear regression model appears to be a little better, which makes sense since the model is considering more relevant factors. However, the second model also appears to underfit the training data. The third model appears to be the best based on R-squared. Based on the residual plots for the first and second models, the relationships between the predictors and the target is not totally linear, so it makes sense that polynomial regression would better fit the training data.

Model Testing

Now, we will evaluate how each model performs against the test data.

```
pred1 <- predict(lm1, newdata=test)
cor1 <- cor(pred1, test$cnt)
mse1 <- mean((pred1-test$cnt)^2)
rmse1 <- sqrt(mse1)
print(paste('correlation of lm1:', cor1))
```

```
## [1] "correlation of lm1: 0.429833704233738"
```

```
print(paste('mse of lm1:', mse1))
```

```
## [1] "mse of lm1: 26002.9650010538"
```

```
print(paste('rmse of lm1:', rmse1))
```

```
## [1] "rmse of lm1: 161.254348781835"
```

```
pred2 <- predict(lm2, newdata=test)
cor2 <- cor(pred2, test$cnt)
mse2 <- mean((pred2-test$cnt)^2)
rmse2 <- sqrt(mse2)
print(paste('correlation of lm2:', cor2))
```

```
## [1] "correlation of lm2: 0.63238736838085"
```

```
print(paste('mse of lm2:', mse2))
```

```
## [1] "mse of lm2: 19114.6961771457"
```

```
print(paste('rmse of lm2:', rmse2))
```

```
## [1] "rmse of lm2: 138.255908290191"
```

```
pred3 <- predict(lm3, newdata=test)
cor3 <- cor(pred3, test$cnt)
mse3 <- mean((pred3-test$cnt)^2)
rmse3 <- sqrt(mse3)
print(paste('correlation of lm3:', cor3))
```

```
## [1] "correlation of lm3: 0.739979170216449"
```

```
print(paste('mse of lm3:', mse3))
```

```
## [1] "mse of lm3: 14420.5961497729"
```

```
print(paste('rmse of lm3:', rmse3))
```

```
## [1] "rmse of lm3: 120.085786626782"
```

We can tell how each model performed based on the correlation and mse values. A model that has a high correlation and low mse is one that performed better. As we predicted based on the R-squared values of the three models, `lm1` performed the worst on the test data, `lm2` performed better, and `lm3` was the best. These results likely happened because the relationship between the factors and the target is not very linear, as demonstrated by the residual plots. This means that a polynomial model had a better chance at accurately modeling the data.