Michael Bernier
DSCI 302 Fall 2020 B
Final Project

<u>Instructions</u>

1. Load the dataset pmsm_temperature_data.xlsx. Preview the document into memory.
2. Consider the following predictors, Ambien, Coolant, u_d, u_q, motor_speed, Torque, stator_yoke, and stator_winding. List the categorical variable from this list and convert it to a factor.
3. Calculate the minimum, maximum, mean, median, standard deviation and three quartiles (25th, 50th and 75th percentiles) of Pm.
4. Calculate the minimum, maximum, mean, median, standard deviation and three quartiles (25th, 50th and 75th percentiles) of motor_speed.
5. Calculate the correlation coefficient of the two variables: motor_speed and Pm. Do they have a strong relationship?
6. Calculate the frequency table of stator_yoke? What's the mode of stator_yoke variable?
7. Plot the histogram and density of the Pm and add the vertical line denoting the mean using ggplot2.
8. Construct the scatter plot of Pm (y-axis) against motor_speed (x-axis) and add the trend line using ggplot2.
9. Plot the boxplot Pm (y-axis) against stator_yoke (x-axis) and save the graph in a file, pmyoke.jpg, using ggplot2. Are there any differences in Pm with respect to stator_yoke?
10. Build the following multiple linear regression models:
    a. Perform multiple linear regression with Pm as the response and the predictors are: Ambien, Coolant, motor_speed, and Torque. Write down the math formula with numerical coefficients.
    b. Perform multiple linear regression with Pm as the response and the predictors are: Ambien, Coolant, u_d, motor_speed, Torque, and stator_winding. Write down the math formula with numerical coefficients.
    c. Perform multiple linear regression with Pm as the response and the predictors are: Ambien, Coolant, u_d, u_q, motor_speed, Torque, stator_yoke, and stator_winding. Write down the math formula with numerical coefficients.
    d. Which model do you recommend to the management based on adjusted R squared? Justify your answer.
11. Build the following KNN models:
    a. Split the data into training dataset (85% of the original data) and test data set (15%)
    b. Forecast stator_yoke using Pm, Ambien, and Coolant.
    c. Forecast the stator_yoke using Pm, Ambien, Coolant, and motor_speed
    d. Forecast the stator_yoke using Pm, Ambien, Coolant, u_d, u_q, motor_speed, and Torque
    e. Which model do you recommend to the management based on accuracy of the test data set? Justify your answer

```
> ##  1. Load the dataset pmsm_temperature_data.xlsx Preview the document into memory.
>
> library(readxl)
> pmsm_temperature_data <- read_excel("pmsm_temperature_data-1.xlsx")
>
> View(pmsm_temperature_data)
>
```

ProjectFinal_Bernier_Michael.R | pmsm_temperature_data

Filter

| | ambient | coolant | u_d | u_q | motor_speed | torque | i_d | i_q | pm | stator_yoke | stator_winding |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -0.58584560 | -0.64553260 | 0.31514500 | -1.250023200 | -1.222435000 | -0.255639730 | 1.02914610 | -0.245720850 | -1.808171400 | Negative | -1.658791900 |
| 2 | -0.49890324 | -0.58888750 | -0.82060474 | 0.083010350 | -0.410784540 | 1.311668200 | 0.14749907 | 1.442951600 | -1.073419000 | Negative | -0.989527940 |
| 3 | 0.04454406 | -0.15780021 | 0.54054374 | -1.184623000 | -1.058541400 | -2.275964000 | -0.18324750 | -2.359318700 | -1.354262500 | Positive | 1.472765600 |
| 4 | 1.16292610 | 0.79740113 | 1.46007180 | 1.127421000 | 0.941319600 | -0.927091500 | -0.33374290 | -0.903193800 | 1.431789900 | Positive | 1.009746200 |
| 5 | -0.20198159 | -1.04870190 | -1.14989240 | 0.649478440 | 1.888843500 | 0.288465440 | -1.28719350 | 0.233622330 | 0.009034172 | Negative | 0.268646900 |
| 6 | 0.68668354 | -0.60714570 | 1.50313830 | -0.101762900 | -0.366283240 | -1.899044200 | 0.14535129 | -2.009588000 | 0.783013400 | Negative | -0.666370500 |
| 7 | -1.21355120 | -1.04107770 | -0.90619564 | 1.020554200 | 2.007889700 | 0.153796200 | -1.09730640 | 0.117008306 | 0.410907480 | Negative | 0.108213566 |
| 8 | -0.09163596 | 0.34515858 | 0.38623545 | 1.732212900 | 1.607866400 | -0.334252400 | -0.46544787 | -0.325432100 | 0.174238670 | Positive | 0.268704120 |
| 9 | 0.69631845 | 0.32776750 | -0.45707867 | 1.508487900 | 1.315254900 | 0.039053623 | -0.44583288 | 0.039120466 | 0.007127435 | Positive | -0.125303100 |
| 10 | 0.49551940 | -0.67226770 | -0.38929520 | 1.498759700 | 1.740211200 | -0.027104175 | -0.70894360 | -0.037400838 | -0.008665779 | Negative | -0.077121090 |
| 11 | -0.13508002 | -1.08223400 | 0.31241680 | -1.250492300 | -1.222431000 | -0.190485280 | 1.06010570 | -0.257236240 | 0.216600550 | Negative | -1.844696900 |
| 12 | -2.74229100 | -1.07347020 | -0.35286380 | 0.836239460 | -0.161884260 | 0.327396630 | 0.85015315 | 0.417112440 | -1.280631800 | Negative | -1.474757100 |
| 13 | 1.42994820 | 1.24556730 | -1.24044750 | 0.560361200 | 1.338915600 | 0.439616700 | -1.00481600 | 0.378724220 | 2.063506000 | Positive | 2.007207200 |
| 14 | 1.18410110 | 1.00827250 | -1.44584450 | 0.017283909 | 1.618296500 | 0.476200200 | -1.48989360 | 0.409843900 | 1.618944000 | Positive | 1.875490800 |

Showing 1 to 15 of 10,000 entries, 11 total columns

```
> ##  2. Consider the following predictors, Ambien, Coolant, u_d, u_q, motor_speed,
> ##     Torque, stator_yoke, and stator_winding.  List the categorical variable from
> ##     this list and convert it to a factor.
>
> ## display all variables and their classifications
> sapply(pmsm_temperature_data, class)
       ambient        coolant            u_d            u_q    motor_speed         torque            i_d            i_q
     "numeric"      "numeric"      "numeric"      "numeric"      "numeric"      "numeric"      "numeric"      "numeric"
            pm    stator_yoke stator_winding
     "numeric"    "character"      "numeric"
>
> ## stator_yoke is the only non-numeric variable; converting to factor
> pmsm_temperature_data$stator_yoke <- as.factor(pmsm_temperature_data$stator_yoke)
>
```

```
> ##  3. Calculate the minimum, maximum, mean, median, standard deviation and three
> ##     quartiles (25th, 50th and 75th percentiles) of Pm.
>
> min(pmsm_temperature_data$pm)
[1] -2.631717
> max(pmsm_temperature_data$pm)
[1] 2.909329
> mean(pmsm_temperature_data$pm)
[1] -0.03546573
> median(pmsm_temperature_data$pm)
[1] 0.04639234
> sd(pmsm_temperature_data$pm)
[1] 1.054456
> quantile(pmsm_temperature_data$pm)
         0%         25%         50%         75%        100% 
-2.63171650 -0.76964945  0.04639234  0.69555275  2.90932870 
>
```

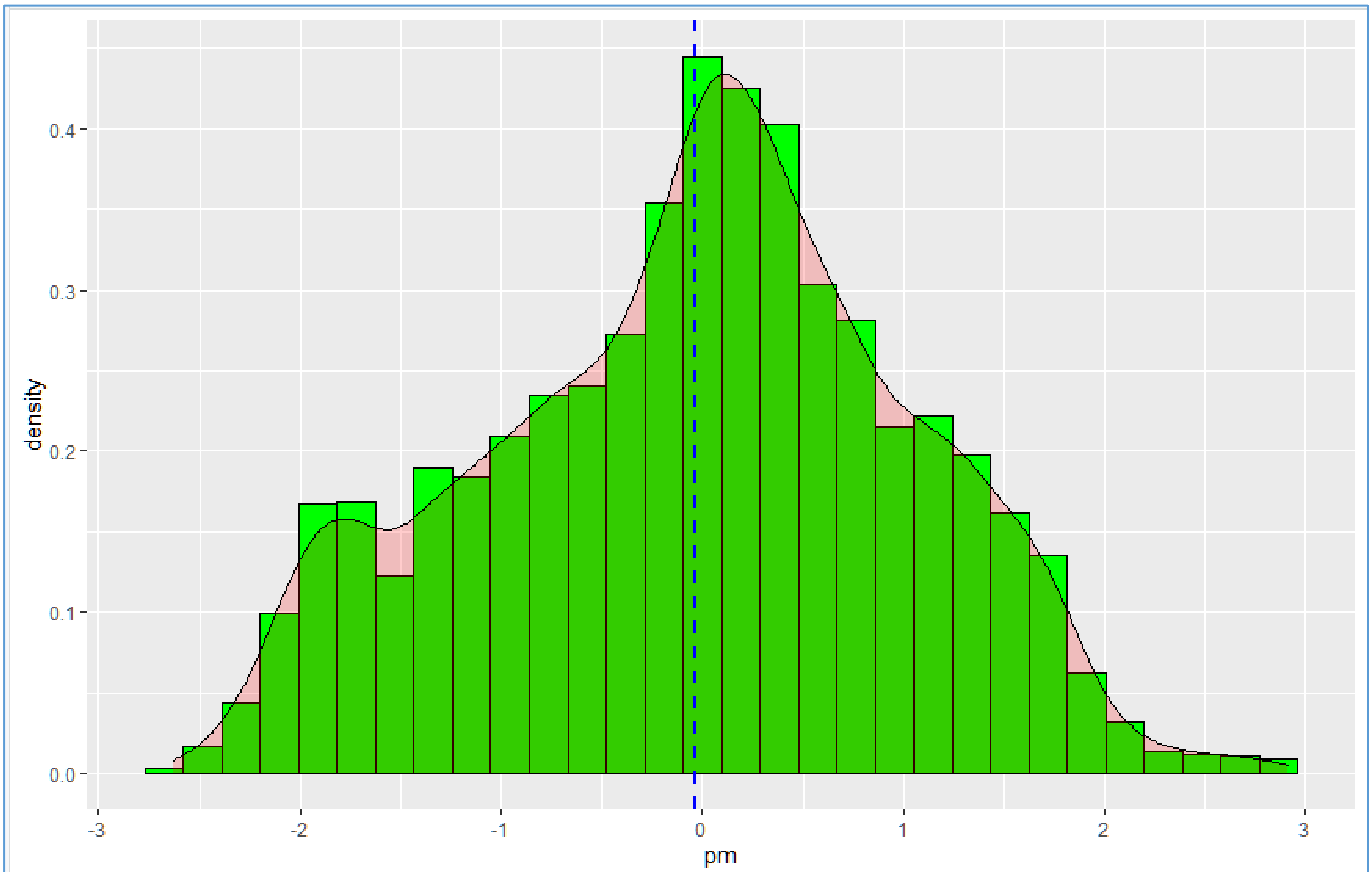
```
> ##  4. Calculate the minimum, maximum, mean, median, standard deviation and three
> ##     quartiles (25th, 50th and 75th percentiles) of motor_speed.
>
> min(pmsm_temperature_data$motor_speed)
[1] -1.222438
> max(pmsm_temperature_data$motor_speed)
[1] 2.024132
> mean(pmsm_temperature_data$motor_speed)
[1] 0.05664231
> median(pmsm_temperature_data$motor_speed)
[1] -0.08494509
> sd(pmsm_temperature_data$motor_speed)
[1] 0.9751519
> quantile(pmsm_temperature_data$motor_speed)
         0%         25%         50%         75%        100% 
-1.22243830 -0.81661443 -0.08494509  0.88356669  2.02413250 
>
```

```
> ##  5. Calculate the correlation coefficient of the two variables: motor_speed and Pm.
>
> cor(pmsm_temperature_data$motor_speed,pmsm_temperature_data$pm)
[1] 0.4085728
>
> ##     Do they have a strong relationship?
>
> ## No, they do not have a strong relationship
>
```

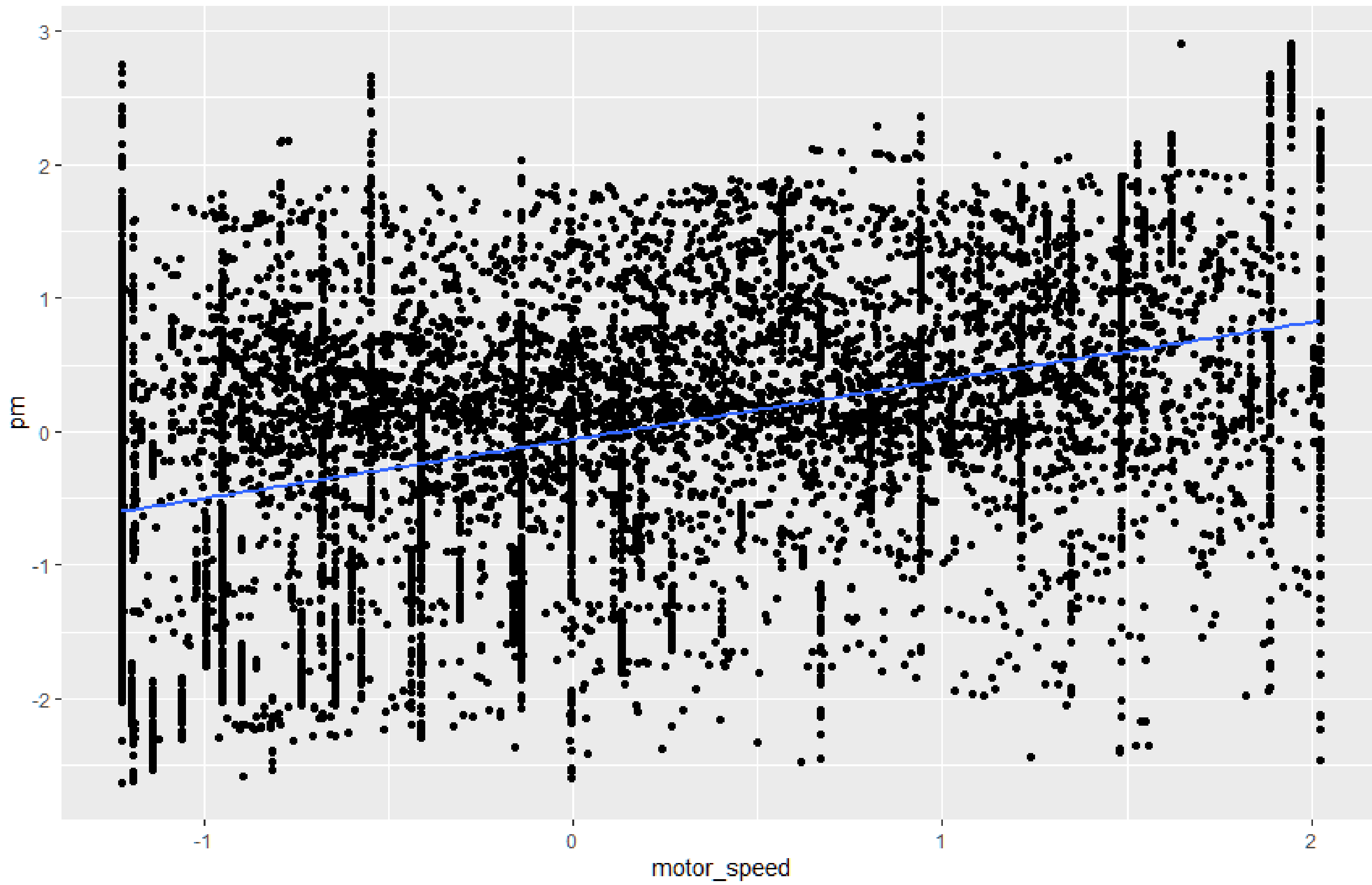
```
> ##  6. Calculate the frequency table of stator_yoke? What's the mode of stator_yoke
> ##     variable?
>
> ## frequency table
> table(pmsm_temperature_data$stator_yoke)

Negative Positive
    5824     4176
>
> ## find mode
> names(sort(-table(pmsm_temperature_data$stator_yoke)))[1]
[1] "Negative"
>
```

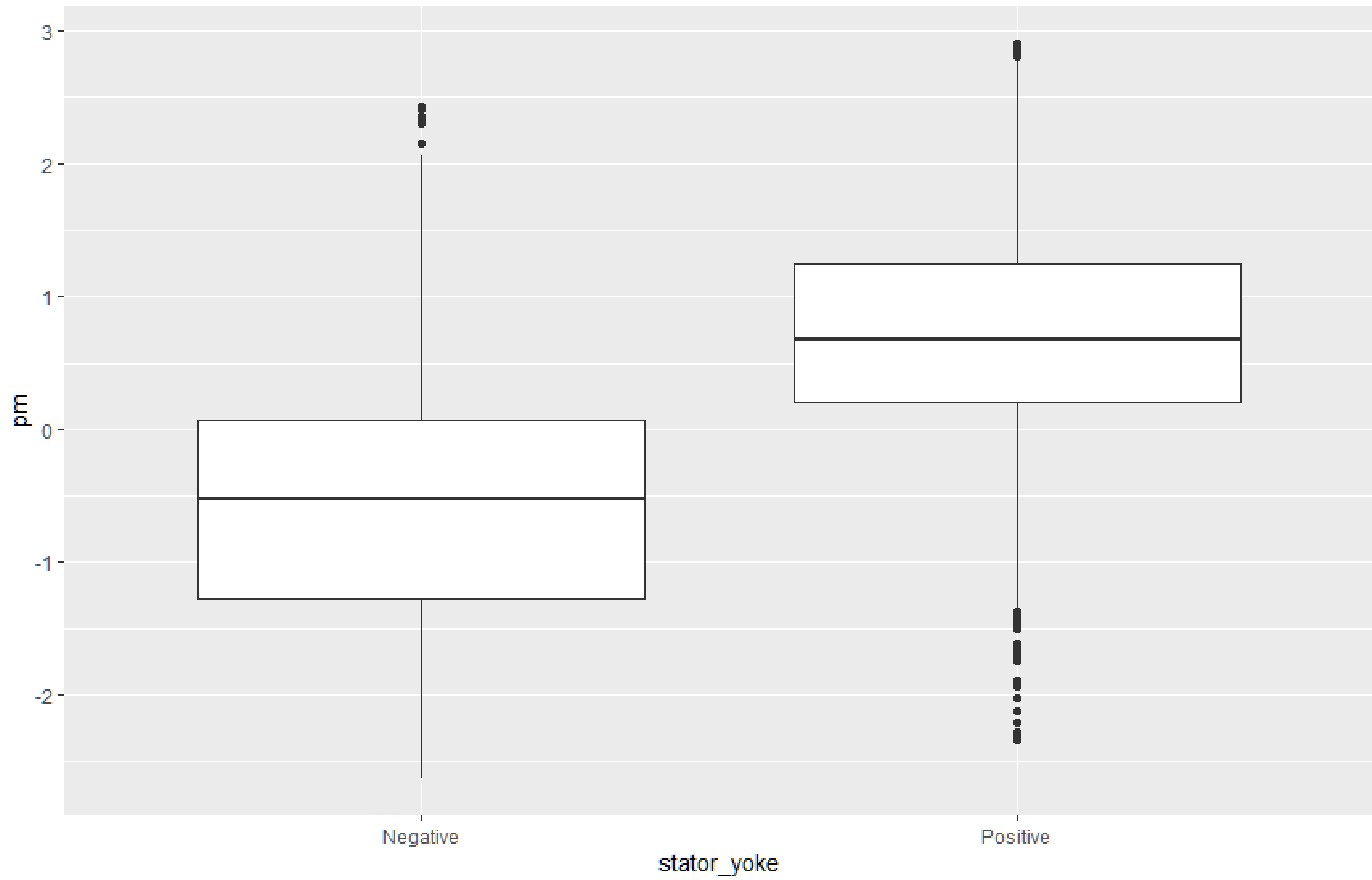
```
> ##  7. Plot the histogram and density of the Pm and add the vertical line denoting
> #       the mean using ggplot2.
>
> library(ggplot2)
>
> ggplot(data = pmsm_temperature_data, aes(x=pm)) +
+   geom_histogram(aes(y=..density..), colour="black", fill="green") +
+   geom_density(alpha = 0.2, fill="red") +
+   geom_vline(aes(xintercept = mean(pm)), color="blue", linetype = "dashed", size=1)
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
>
```



```
> ##  8. Construct the scatter plot of Pm (y-axis) against motor_speed (x-axis) and
> ##     add the trend line using ggplot2.
>
> ggplot(data = pmsm_temperature_data, aes(x=motor_speed, y=pm))+
+   geom_point() +
+   geom_smooth(method="lm", se=FALSE)
`geom_smooth()` using formula 'y ~ x'
>
```



```
> ##  9. Plot the boxplot Pm (y-axis) against stator_yoke (x-axis) and save the graph
> ##     in a file, pmyoke.jpg, using ggplot2. Are there any differences in Pm with
> ##     respect to stator_yoke?
>
> ggplot(data = pmsm_temperature_data, aes(x=stator_yoke, y=pm))+
+   geom_boxplot()
> ggsave("pmyoke.jpg")
Saving 9.02 x 5.78 in image
>
>
> ## Yes. The values for Pm are in the negative range when the stator_yoke value is
> ## "Negative" and in the positive range when the stator_yoke value is "Positive".
>
```



```
> ## 10. Build the following multiple linear regression models:
>
> ##     a. Preform multiple linear regression with Pm as the response and the predictors
> ##        are: Ambien, Coolant, motor_speed, and Torque. Write down the math formula
> ##        with numerical coefficients.
>
> lm.result1 <- lm(pm ~ ambient + coolant + motor_speed + torque, data=pmsm_temperature_data)
> summary(lm.result1)

Call:
lm(formula = pm ~ ambient + coolant + motor_speed + torque, data = pmsm_temperature_data)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6782 -0.4846 -0.0646  0.4619  3.4072

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 0.009985   0.007962   1.254     0.21
ambient     0.345500   0.008618  40.092   <2e-16 ***
coolant     0.302666   0.009310  32.511   <2e-16 ***
motor_speed 0.398226   0.008104  49.141   <2e-16 ***
torque      0.072800   0.008078   9.013   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.783 on 9995 degrees of freedom
Multiple R-squared:  0.4488,    Adjusted R-squared:  0.4486
F-statistic:  2035 on 4 and 9995 DF,  p-value: < 2.2e-16

>
```

Math formula:
pm = 0.009985 + (0.345500 * ambient) + (0.302666 * coolant) + (0.398226 * motor_speed) + (0.072800 * torque)

```
> ##     b. Preform multiple linear regression with Pm as the response and the predictors
> ##        are: Ambien, Coolant, u_d,  motor_speed, Torque, and stator_winding. Write down
> ##        the math formula with numerical coefficients.
>
> lm.result2 <- lm(pm ~ ambient + coolant + u_d + motor_speed + torque + stator_winding,
+                  data=pmsm_temperature_data)
> summary(lm.result2)

Call:
lm(formula = pm ~ ambient + coolant + u_d + motor_speed + torque +
    stator_winding, data = pmsm_temperature_data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.96754 -0.36978 -0.01535  0.34878  2.42477

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     0.006438   0.006183   1.041   0.2978
ambient         0.282715   0.006740  41.946  < 2e-16 ***
coolant        -0.014670   0.008365  -1.754   0.0795 .
u_d            -0.265500   0.011479 -23.130  < 2e-16 ***
motor_speed     0.048006   0.007873   6.098 1.11e-09 ***
torque         -0.289061   0.011282 -25.621  < 2e-16 ***
stator_winding  0.618445   0.007950  77.793  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6081 on 9993 degrees of freedom
Multiple R-squared:  0.6676,    Adjusted R-squared:  0.6674
F-statistic:  3346 on 6 and 9993 DF,  p-value: < 2.2e-16

>
```

Math formula:
pm = 0.006438 + (0.282715 * ambient) + (-0.014670 * coolant) + (-0.265500 * u_d) + (0.048006 * motor_speed) + (-0.289061 * torque) + (0.618445 * stator_winding)

```
> ##     c. Preform multiple linear regression with Pm as the response and the predictors
> ##        are: Ambien, Coolant, u_d, u_q, motor_speed, Torque, stator_yoke, and
> ##        stator_winding. Write down the math formula with numerical coefficients.
>
> lm.result3 <- lm(pm ~ ambient + coolant + u_d + u_q + motor_speed + torque + stator_winding
+                  + stator_yoke, data = pmsm_temperature_data)
> summary(lm.result3)

Call:
lm(formula = pm ~ ambient + coolant + u_d + u_q + motor_speed +
    torque + stator_winding + stator_yoke, data = pmsm_temperature_data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.94803 -0.36849 -0.01985  0.35046  2.42405

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         -0.123766   0.011723 -10.558  < 2e-16 ***
ambient              0.286867   0.006692  42.865  < 2e-16 ***
coolant             -0.067238   0.009891  -6.798 1.12e-11 ***
u_d                 -0.247135   0.011862 -20.834  < 2e-16 ***
u_q                 -0.057886   0.009610  -6.024 1.77e-09 ***
motor_speed          0.106641   0.011784   9.050  < 2e-16 ***
torque              -0.264481   0.011596 -22.808  < 2e-16 ***
stator_winding       0.530634   0.010112  52.478  < 2e-16 ***
stator_yokePositive  0.284502   0.021421  13.282  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6021 on 9991 degrees of freedom
Multiple R-squared:  0.6742,    Adjusted R-squared:  0.674
F-statistic:  2585 on 8 and 9991 DF,  p-value: < 2.2e-16

>
```

Math formulas:

pm = 0.160736 + (0.286867 * ambient) + (-0.067238 * coolant) + (0.247135 * u_d) + (-0.057886 * u_q) + (0.106641 * motor_speed) + (-0.264481 * torque) + (0.530634 * stator_winding)  **[stator_yoke is Positive]**

pm = -0.123766 + (0.286867 * ambient) + (-0.067238 * coolant) + (0.247135 * u_d) + (-0.057886 * u_q) + (0.106641 * motor_speed) + (-0.264481 * torque) + (0.530634 * stator_winding)  **[stator_yoke is Negative]**

Note: first equation has been simplified by adding together the values for the intercept and the coefficient for stator_yokePositive (-0.123766 + 0.284502 respectively)

```
> ##     d. Which model do you recommend to the management based on adjusted R squared?
> ##        Justify your answer.
>
> ## I recommend model (c) because its adjusted R squared value is the highest (0.674
> ## compared to 0.6674 for model (b) and 0.4486 for model (a). The higher R squared
> ## value indicates model (c) provides the best fit for the data.
>
```

```
> ## 11. Build the following KNN models:
>
> ##     a. split the data into training dataset (85% of the original data) and test
> ##        data set (15%)
>
> ## load class library
> library(class)
>
> ## play it safe and normalize all of the data
>
> ## create normalizing function
> NormalizedData <- function(vinput) {
+   result <- (vinput - min(vinput))/(max(vinput) - min(vinput))
+   return(result)
+ }
>
> ## normalizing data
> pmsm_temperature_data$pm          <- NormalizedData(pmsm_temperature_data$pm)
> pmsm_temperature_data$ambient     <- NormalizedData(pmsm_temperature_data$ambient)
> pmsm_temperature_data$coolant     <- NormalizedData(pmsm_temperature_data$coolant)
> pmsm_temperature_data$motor_speed <- NormalizedData(pmsm_temperature_data$motor_speed)
> pmsm_temperature_data$torque      <- NormalizedData(pmsm_temperature_data$torque)
> pmsm_temperature_data$u_d         <- NormalizedData(pmsm_temperature_data$u_d)
> pmsm_temperature_data$u_q         <- NormalizedData(pmsm_temperature_data$u_q)
>
> ## set training sample to 85% of dataset
> train.sample <- floor(0.85 * nrow(pmsm_temperature_data))
>
```

```
> ##     b. forecast stator_yoke using Pm, Ambien, and Coolant.
>
> ## set target for forecast
> fcast1.target <- pmsm_temperature_data$stator_yoke
>
> ## set predictors
> predictors1 <- c("pm", "ambient", "coolant")
>
> ## select predictors
> fcast1.predictors <- pmsm_temperature_data[predictors1]
>
> ## generate training model
> fcast1.train <- fcast1.predictors[1:train.sample, ]
>
> ## generate testing model
> fcast1.test <- fcast1.predictors[-c(1:train.sample), ]
>
> ## select corresponding labels
> fcast1.cl <- fcast1.target[1:train.sample]
>
> ## computer number of neighbors
> fcast1.neighbors <- floor(sqrt(nrow(pmsm_temperature_data)))
>
> ## run KNN algorithm
> fcast1.knn.predict <- knn(fcast1.train, fcast1.test, fcast1.cl, k = fcast1.neighbors)
>
> ## confusion matrix/contingency table
> fcast1.label <- fcast1.target[-c(1:train.sample)]
> table(fcast1.label, fcast1.knn.predict)
            fcast1.knn.predict
fcast1.label Negative Positive
    Negative      849       50
    Positive       72      529
>
```

```
> ##     c. forecast the stator_yoke using Pm, Ambien, Coolant, and  motor_speed
>
> ## set target for forecast
> fcast2.target <- pmsm_temperature_data$stator_yoke
>
> ## set predictors
> predictors2 <- c("pm", "ambient", "coolant", "motor_speed")
>
> ## select predictors
> fcast2.predictors <- pmsm_temperature_data[predictors2]
>
> ## generate training model
> fcast2.train <- fcast2.predictors[1:train.sample, ]
>
> ## generate testing model
> fcast2.test <- fcast2.predictors[-c(1:train.sample), ]
>
> ## select corresponding labels
> fcast2.cl <- fcast2.target[1:train.sample]
>
> ## computer number of neighbors
> fcast2.neighbors <- floor(sqrt(nrow(pmsm_temperature_data)))
>
> ## run KNN algorithm
> fcast2.knn.predict <- knn(fcast2.train, fcast2.test, fcast2.cl, k = fcast2.neighbors)
>
> ## confusion matrix/contingency table
> fcast2.label <- fcast2.target[-c(1:train.sample)]
> table(fcast2.label, fcast2.knn.predict)
            fcast2.knn.predict
fcast2.label Negative Positive
    Negative      862       37
    Positive       77      524
>
```

```
> ##     d. forecast the stator_yoke using Pm, Ambien, Coolant, u_d, u_q, motor_speed,
> ##        and Torque
>
> ## set target for forecast
> fcast3.target <- pmsm_temperature_data$stator_yoke
>
> ## set predictors
> predictors3 <- c("pm", "ambient", "coolant", "motor_speed", "u_d", "u_q", "torque")
>
> ## select predictors
> fcast3.predictors <- pmsm_temperature_data[predictors3]
>
> ## generate training model
> fcast3.train <- fcast3.predictors[1:train.sample, ]
>
> ## generate testing model
> fcast3.test <- fcast3.predictors[-c(1:train.sample), ]
>
> ## select corresponding labels
> fcast3.cl <- fcast3.target[1:train.sample]
>
> ## computer number of neighbors
> fcast3.neighbors <- floor(sqrt(nrow(pmsm_temperature_data)))
>
> ## run KNN algorithm
> fcast3.knn.predict <- knn(fcast3.train, fcast3.test, fcast3.cl, k = fcast3.neighbors)
>
> ## confusion matrix/contingency table
> fcast3.label <- fcast3.target[-c(1:train.sample)]
> table(fcast3.label, fcast3.knn.predict)
            fcast3.knn.predict
fcast3.label Negative Positive
    Negative      859       40
    Positive       74      527
>
```

```
>
> ##     e. Which model do you recommend to the management based on accuracy of the
> ##        test data set? Justify your answer
>
> ## accuracy of model (b) = (849+528)/(849+50+73+528) = 1377/1500 = 0.918 = 91.8%
> ## accuracy of model (c) = (862+524)/(862+37+77+524) = 1386/1500 = 0.924 = 92.4%
> ## accuracy of model (d) = (858+525)/(858+41+76+525) = 1383/1500 = 0.922 = 92.2%
>
> ## Based on the accuracy of the test data sets, my recommendation would be model (c).
> ## While normally adding more predictors would be expected to improve accuracy, in this case
> ## the addition of u_d, u_q, and torque caused the accuracy of model (d) to decrease slightly,
> ## meaning those values were contradictory to the patterns found in the other predictors
> 
```