Linear Regression Coding Assignment-1

Code ▾

Hide

```
# Load essential libraries
library(ggplot2)
library(dplyr)
```

```

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

Hide

```
library(HSAUR)
```

```
Warning: package ‘HSAUR’ was built under R version 4.3.2Loading required package: tools
```

Hide

```
library(ggcorrplot)
```

```
Warning: package ‘ggcorrplot’ was built under R version 4.3.2
```

Hide

```
# Load the heptathlon dataset
data(heptathlon)
str(heptathlon)
```

```
'data.frame':	25 obs. of  8 variables:
 $ hurdles : num  12.7 12.8 13.2 13.6 13.5 ...
 $ highjump: num  1.86 1.8 1.83 1.8 1.74 1.83 1.8 1.8 1.83 1.77 ...
 $ shot    : num  15.8 16.2 14.2 15.2 14.8 ...
 $ run200m : num  22.6 23.6 23.1 23.9 23.9 ...
 $ longjump: num  7.27 6.71 6.68 6.25 6.32 6.33 6.37 6.47 6.11 6.28 ...
 $ javelin : num  45.7 42.6 44.5 42.8 47.5 ...
 $ run800m : num  129 126 124 132 128 ...
 $ score   : int  7291 6897 6858 6540 6540 6411 6351 6297 6252 6252 ...
```

Hide

```
# Introduce a new column called sprint highlighting slow and fast sprinters
heptathlon = heptathlon %>% mutate(sprint = ifelse(run200m <= 25 & run800m <= 129, 'fast', 'slow'))
str(heptathlon)
```

```
'data.frame':	25 obs. of  9 variables:
 $ hurdles : num  12.7 12.8 13.2 13.6 13.5 ...
 $ highjump: num  1.86 1.8 1.83 1.8 1.74 1.83 1.8 1.8 1.83 1.77 ...
 $ shot    : num  15.8 16.2 14.2 15.2 14.8 ...
 $ run200m : num  22.6 23.6 23.1 23.9 23.9 ...
 $ longjump: num  7.27 6.71 6.68 6.25 6.32 6.33 6.37 6.47 6.11 6.28 ...
 $ javelin : num  45.7 42.6 44.5 42.8 47.5 ...
 $ run800m : num  129 126 124 132 128 ...
 $ score   : int  7291 6897 6858 6540 6540 6411 6351 6297 6252 6252 ...
 $ sprint  : chr  "fast" "fast" "fast" "slow" ...
```

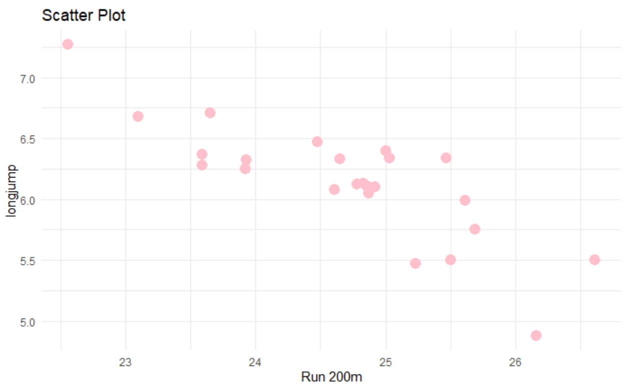
Hide

```
# Change sprint column to factor type
heptathlon['sprint'] = lapply(heptathlon['sprint'], as.factor)
str(heptathlon)
```

```
'data.frame':	25 obs. of  9 variables:
 $ hurdles : num  12.7 12.8 13.2 13.6 13.5 ...
 $ highjump: num  1.86 1.8 1.83 1.8 1.74 1.83 1.8 1.8 1.83 1.77 ...
 $ shot    : num  15.8 16.2 14.2 15.2 14.8 ...
 $ run200m : num  22.6 23.6 23.1 23.9 23.9 ...
 $ longjump: num  7.27 6.71 6.68 6.25 6.32 6.33 6.37 6.47 6.11 6.28 ...
 $ javelin : num  45.7 42.6 44.5 42.8 47.5 ...
 $ run800m : num  129 126 124 132 128 ...
 $ score   : int  7291 6897 6858 6540 6540 6411 6351 6297 6252 6252 ...
 $ sprint  : Factor w/ 2 levels "fast","slow": 1 1 1 2 1 1 2 2 2 2 ...
```

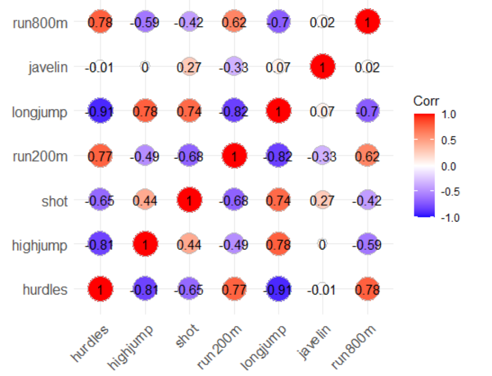
Hide

```
# Make a scatter plot between *run200m* (x-axis) and *longjump* (y-axis). What do you observe from this plot?
p = ggplot(heptathlon, aes(x=run200m,y=longjump))+
  geom_point(color='pink',size=4)+  labs(title = "Scatter Plot",x='Run 200m',y='longjump')+
  theme_minimal()
p
```



Hide

```
# Correlation between all pairs of continuous predictors (leave out sprint and the response variable score). What do you obs
erve?
cor_matrix = cor(heptathlon %>% select(-c(sprint, score)))
ggcorrplot(cor_matrix, method = 'circle', lab = TRUE)
```

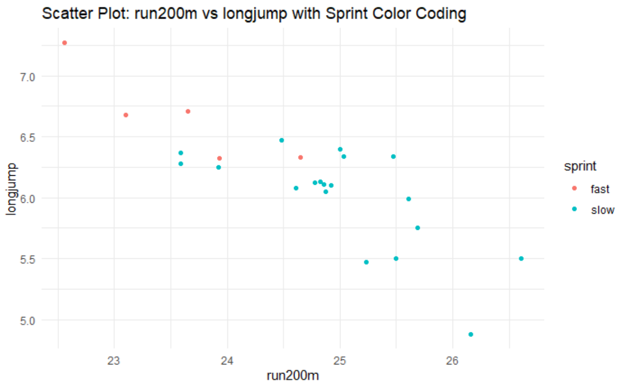


Hide

```
#Values close to 1 indicate a strong positive correlation, while values close to -1 indicate a strong negative correlation.
```

Hide

```
# Make a scatter plot between *run200m* (x-axis) and *longjump* (y-axis) now with the data points color-coded using *sprint
*. What do you observe from this plot?
ggplot(heptathlon, aes(x = run200m, y = longjump, color = sprint)) +
  geom_point() +
  labs(title = "Scatter Plot: run200m vs longjump with Sprint Color Coding",
       x = "run200m", y = "longjump") + theme_minimal()
```



Hide

```
# Calculate Pearson's correlation between *run200m* and *longjump*. What do you observe?

cor2 = cor(heptathlon['run200m'], heptathlon['longjump'], method = "pearson")
cor2
```

```
          longjump
run200m -0.8172053
```

Hide

```
# How many levels does the categorical variable *sprint* have? What is the reference level?
contrasts(heptathlon$sprint)
```

```
     slow
fast    0
slow    1
```

Hide

```
levels(heptathlon$sprint)
```

```
[1] "fast" "slow"
```

Hide

```
# Fit a linear model for approximating *score* as a function of *sprint*. Print the model's summary. How accurate is the mod
el? How do the slow athletes' scores compare to the fast ones?
model = lm(data = heptathlon, score ~ sprint)
summary(model)
```

```
Call:
lm(formula = score ~ sprint, data = heptathlon)

Residuals:
    Min      1Q  Median      3Q     Max 
-1347.4  -227.4    97.6   291.6   626.6 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   6799.4      200.3  33.939  < 2e-16 ***
sprintslow    -886.0      224.0  -3.956 0.000628 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 448 on 23 degrees of freedom
Multiple R-squared:  0.4049,	Adjusted R-squared:  0.379 
F-statistic: 15.65 on 1 and 23 DF,  p-value: 0.0006282
```

Hide

```
mean_slow = mean(heptathlon[heptathlon$sprint == 'slow', 'score'])
mean_fast = mean(heptathlon[heptathlon$sprint == 'fast', 'score'])
mean_slow
```

```
[1] 5913.4
```

Hide

```
mean_fast
```

```
[1] 6799.4
```

Hide

```
mean_slow-mean_fast
```

```
[1] -886
```

Hide

```
# Fit a linear model for approximating *score* as a function of *shot* and *sprint*. Print the model's summary and answer th
e following questions:

# 1. Did the addition of the new predictor *shot* improve the model accuracy?
# 2. *True/false* (explain in one line): the model suggests that there is a possible linear relationship between an athlet
e's score and shotput performance.
# 3. For a 1 metre increase in shot put throw and with the same sprint performance, we can say with 95% confidence that the
athlete's score will increase/decrease by an amount in the interval [?, ?].
model = lm(data = heptathlon,score ~ shot + sprint)
summary(model)
```

```

Call:
lm(formula = score ~ shot + sprint, data = heptathlon)

Residuals:
     Min       1Q   Median       3Q      Max 
-1124.58  -164.40    35.93   207.34   496.35 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   3080.0      883.0   3.488 0.002084 ** 
shot           249.7       58.4   4.275 0.000308 ***
sprintslow    -330.4      213.4  -1.548 0.135842    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 338.5 on 22 degrees of freedom
Multiple R-squared:  0.6749,	Adjusted R-squared:  0.6454 
F-statistic: 22.84 on 2 and 22 DF,  p-value: 4.282e-06
```

Hide

```
#  Using the model built above, extract the slope and intercept for estimating the *score* of *slow* and *fast* athletes.
# For slow athletes
intercept_slow = 3080.0
slope_slow = 249.7

# For fast athletes
intercept_fast = 2749.6
slope_fast = 249.7
```

Hide

```
# Complete the code below to build a linear model for approximating *score* as a function of *shot* and *sprint* using the t
raining data. Predict the model performance by applying it to the test data.
# Split the data into 80% train and 20% test parts
set.seed(0)
train_ind = sample(1:nrow(heptathlon), size = 0.8*nrow(heptathlon))

hDataTrain = heptathlon[train_ind, ]
hDataTest = heptathlon[-train_ind, ]

# Build linear regression model
model = lm(score ~ shot + sprint, data = hDataTrain)

# Predict on the test data
predictions = predict(model, newdata = hDataTest)

# Print the true and predicted scores for the test data
print(cbind(TrueScore = hDataTest$score, PredictedScore = predictions))
```

```
               TrueScore PredictedScore
Behmer (GDR)        6858       6549.446
Greiner (USA)       6297       6279.790
Scheider (SWI)      6137       5592.081
Kytola (FIN)        5686       5613.656
Jeong-Mi (KOR)      5289       5389.814
```

Hide

```
# Calculate the model error (mean-squared error for test data)
mse = mean((hDataTest$score - predictions)^2)
print(paste("Mean Squared Error: ", mse))
```

```
[1] "Mean Squared Error:  81567.1356660685"
```

Hide

```
# Fit a linear model for approximating *score* as a function of *shot*, *javelin*, and *sprint*. Print the model's summary a
nd answer the following questions:

#1. Did the addition of the new predictor *javelin* improve the model accuracy?
#2. *True/false* (explain in one line): the model suggests that there is a possible linear relationship between an athlete's
score and javelin performance.
#3. For a 1 metre increase in shot put throw and with the same javelin and sprint performance, we can say with 95% confidenc
e that the athlete's score will increase/decrease by an amount in the interval [?, ?].
model =  lm(score ~ shot + javelin + sprint, data = hDataTrain)
summary(model)
```

```
Call:
lm(formula = score ~ shot + javelin + sprint, data = hDataTrain)

Residuals:
    Min      1Q  Median      3Q     Max 
-908.76 -113.96    8.78  204.83  449.76 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)   
(Intercept)  3860.20    1521.11   2.538  0.02194 * 
shot          277.65      71.55   3.880  0.00133 **
javelin       -28.24      26.35  -1.072  0.29966   
sprintslow   -346.28     269.83  -1.283  0.21766   
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 354.9 on 16 degrees of freedom
Multiple R-squared:  0.6808,	Adjusted R-squared:  0.621 
F-statistic: 11.38 on 3 and 16 DF,  p-value: 0.0003043
```

Hide

```
# Fit a linear model for approximating *score* as a function of *highjump*, and *sprint*. Print the model's summary and answ
er the following questions:
# 1. How accurate is this model?
# 2. Considering a p-value of 10% as cutoff, are there any insignificant features?
model = lm(data = hDataTrain,score ~ highjump + sprint)
summary(model)
```

```
Call:
lm(formula = score ~ highjump + sprint, data = hDataTrain)

Residuals:
    Min      1Q  Median      3Q     Max 
-481.77 -156.51  -10.06  122.85  476.28 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -1893.2     1251.8  -1.512 0.148801    
highjump      4801.1      689.2   6.966 2.28e-06 ***
sprintslow    -685.0      139.8  -4.902 0.000135 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 246.1 on 17 degrees of freedom
Multiple R-squared:  0.8369,	Adjusted R-squared:  0.8177 
F-statistic: 43.62 on 2 and 17 DF,  p-value: 2.02e-07
```