Linear Regression Coding Assignment-1

```
# 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
```

```
library(HSAUR)
```

```
## Warning: package 'HSAUR' was built under R version 4.3.2
```

```
## Loading required package: tools
```

```
library(ggcorrplot)
```

```
## Warning: package 'ggcorrplot' was built under R version 4.3.2
```

```
# 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 ...
```

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

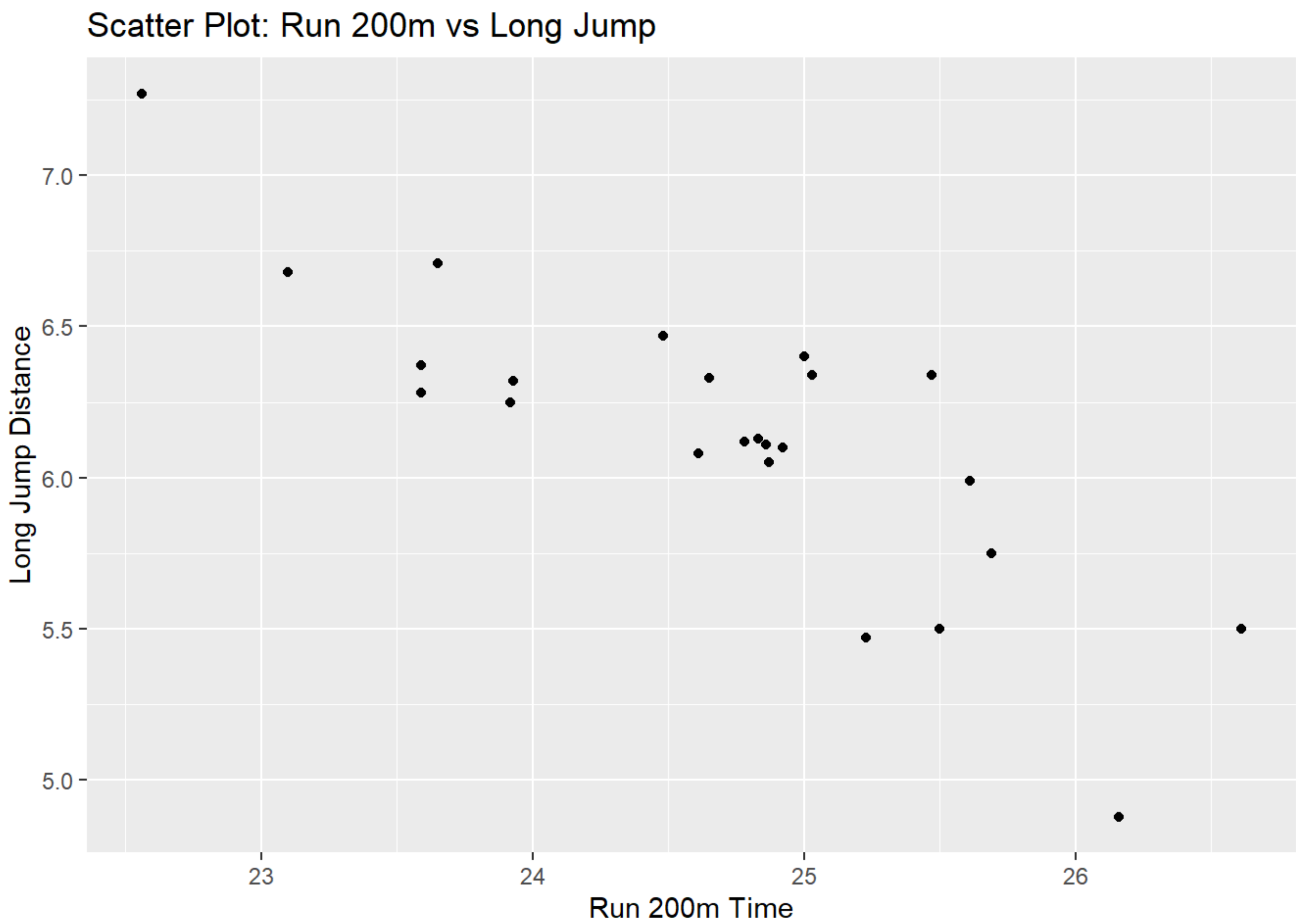
```
## 'data.frame':    25 obs. of  9 variables:
##  $ hurdles : num  12.7 12.8 13.2 13.6 13.5 ...
##  $ highjump: num  1.86 1.8 1.83 1.8 1.74 1.83 1.8 1.8 1.83 1.77 ...
##  $ shot    : num  15.8 16.2 14.2 15.2 14.8 ...
##  $ run200m : num  22.6 23.6 23.1 23.9 23.9 ...
##  $ longjump: num  7.27 6.71 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" ...
```

```
# Change sprint column to factor type
heptathlon['sprint'] = factor(heptathlon$sprint)
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 ...
```

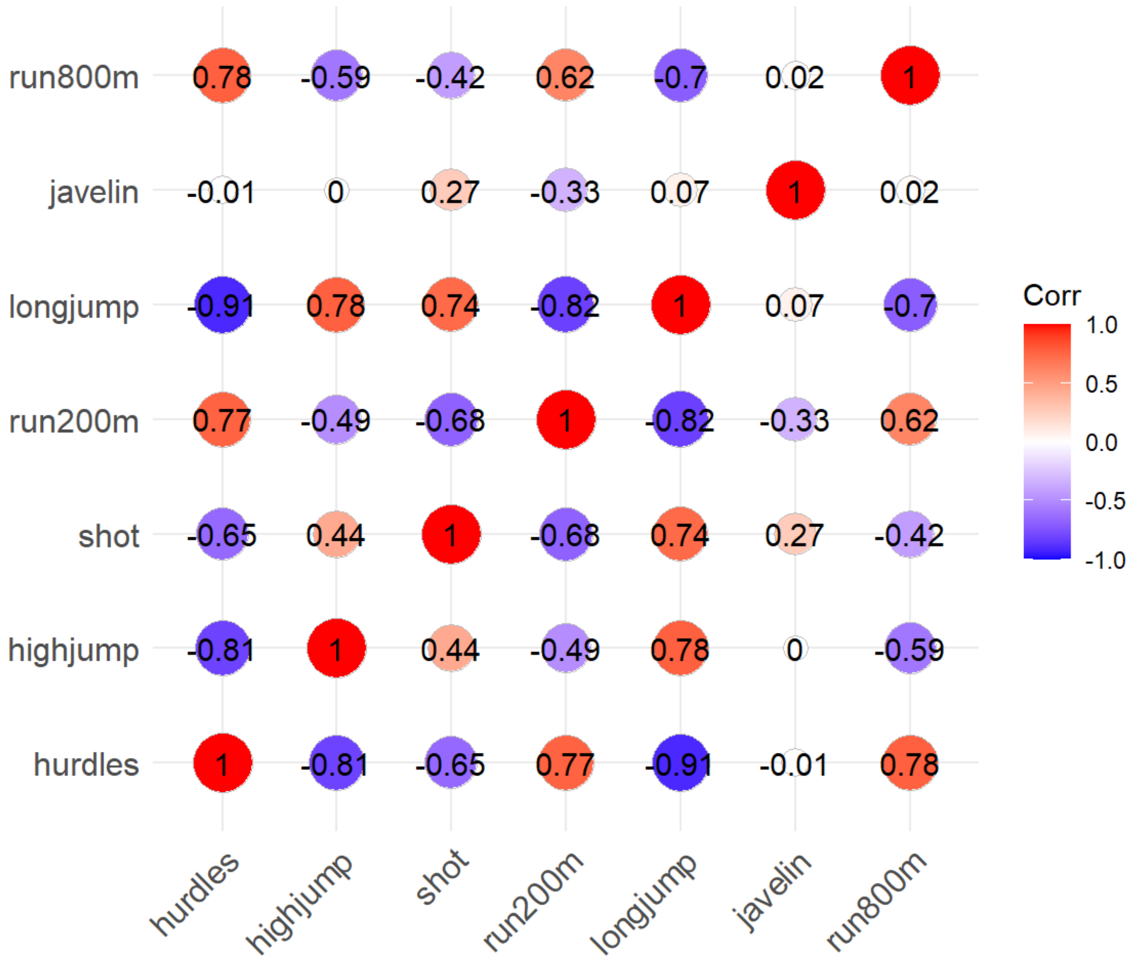
```
# Make a scatter plot between *run200m* (x-axis) and *longjump* (y-axis). What do you obse
rve from this plot?

p =ggplot(heptathlon, aes(x = run200m, y = longjump)) +
  geom_point() +
  labs(x = "Run 200m Time", y = "Long Jump Distance", title = "Scatter Plot: Run 200m vs L
ong Jump")
p
```



from the plot it seems like when the athlete is faster (run time is slow), the long jump distance is more, this suggests correlation between the two is negative, like fast runners can jump long and better

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



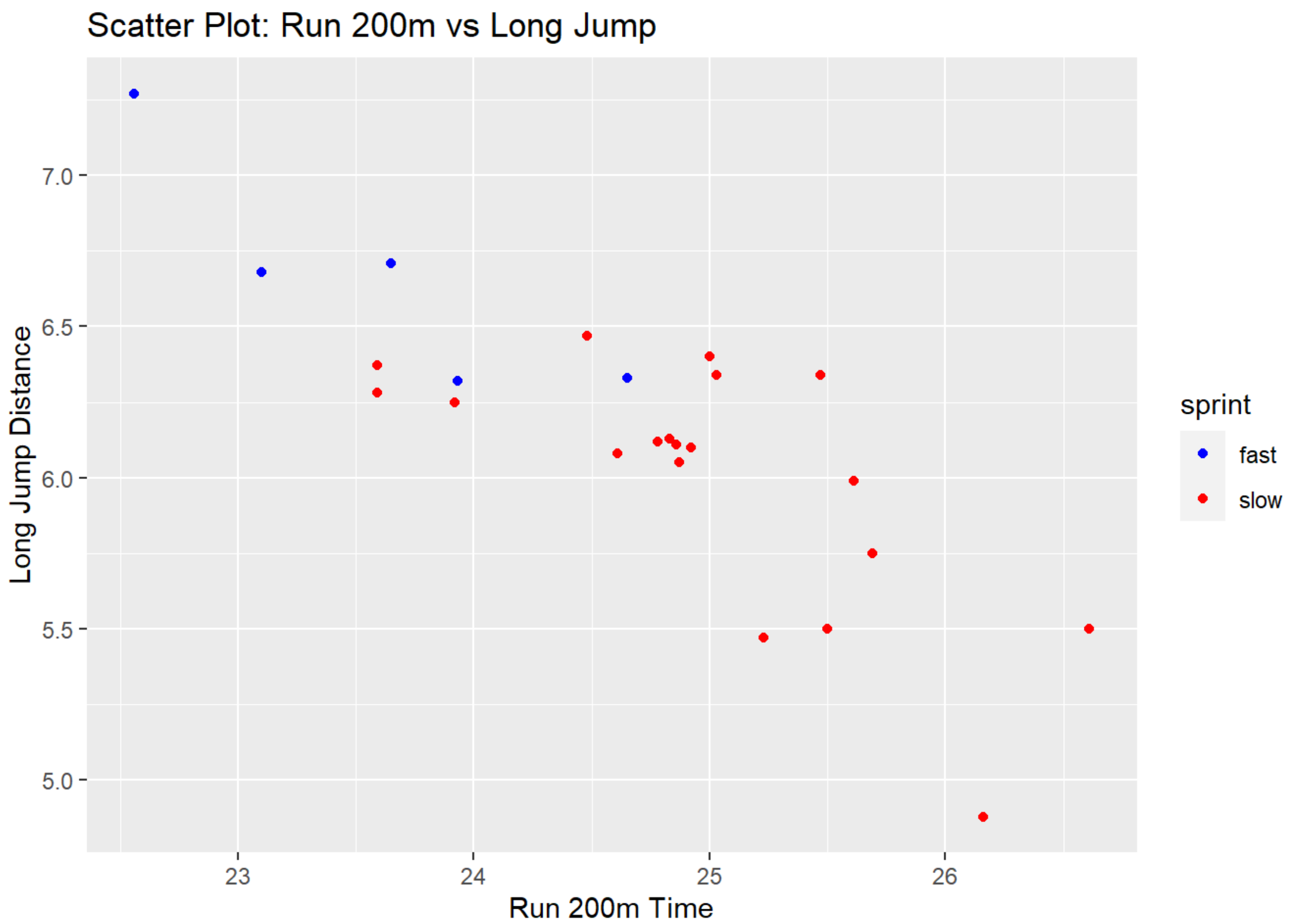
There is highest negative correlation between hurdles and long jump, suggesting as hurdles increase, distance covered in long jump is less

there is significant correlation between long jump and high jump suggesting, athele good in one, can also perform well in the other

there is also significant negative correlation between long jump and run time as observed from previous plot

```
# 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?

p = ggplot(heptathlon, aes(x = run200m, y = longjump, color = sprint)) +
  geom_point() +
  labs(x = "Run 200m Time", y = "Long Jump Distance", title = "Scatter Plot: Run 200m vs L
ong Jump") +
  scale_color_manual(values = c('fast' = 'blue', 'slow' = 'red'))
p
```



_ as understood from before, slow runners take more time and hence cannot jump long distances, but fast runners who take less time can cover more distance in long jump

```
# Calculate Pearson's correlation between *run200m* and *longjump*. What do you observe?
cor(heptathlon$run200m, heptathlon$longjump)
```

```
## [1] -0.8172053
```

there is high negative correlation of -0.81 suggesting the relationship between them as explaine from before

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

```
##      slow
## fast    0
## slow    1
```

2 levels and fast is the reference level as it comes first in alphabetical order

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

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

```
## [1] 6799.4
```

```
print(mean_slow)
```

```
## [1] 5913.4
```

```
print(mean_slow-mean_fast)
```

```
## [1] -886
```

The model is not that good as it’s r-square and adjusted 4 sqaure values are less fast athletes’ scores are comparatively more than slow athletes scores

```
# Fit a linear model for approximating *score* as a function of *shot* and *sprint*. Print the model's summary and answer the 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 athlete'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(score ~ shot + sprint, data = heptathlon)
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
```

```
initial_model = lm(score ~ sprint, data = heptathlon)
improved_accuracy = (summary(model)$r.squared > summary(initial_model)$r.squared)
print(paste("Adding 'shot' improved model accuracy:", improved_accuracy))
```

```
## [1] "Adding 'shot' improved model accuracy: TRUE"
```

```
linear_relationship_shot = summary(model)$coefficients["shot", "Pr(>|t|)"] < 0.05
print(paste("The model suggests a possible linear relationship between an athlete's score
and shotput performance:", linear_relationship_shot))
```

```
## [1] "The model suggests a possible linear relationship between an athlete's score and s
hotput performance: TRUE"
```

```
confidence_interval_shot = confint(model)[2, ]
print(paste("95% Confidence Interval for the effect of shot on score:", confidence_interva
l_shot))
```

```
## [1] "95% Confidence Interval for the effect of shot on score: 128.55515402802"
## [2] "95% Confidence Interval for the effect of shot on score: 370.765217377499"
```

Yes, the addition of the predictor shot improved the model accuracy.

True. The p-value associated with the coefficient for the shot variable is less than 0.05,

The 95% Confidence Interval for the effect of shot on score is [128.56, 370.77].

```
#  Using the model built above, extract the slope and intercept for estimating the *score*
of *slow* and *fast* athlete.
model = lm(score ~ shot + sprint, data = heptathlon)
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
```

```
coefficients = coef(model)
intercept_slow = coefficients["(Intercept)"] + coefficients["sprintslow"]
intercept_fast = coefficients["(Intercept)"]

slope_slow = coefficients["shot"] + coefficients["sprintsprintslow"]
slope_fast = coefficients["shot"] + coefficients["sprintsprintfast"]

print(intercept_slow)
```

```
## (Intercept)
##    2749.581
```

```
print(intercept_fast)
```

```
## (Intercept)
##    3079.963
```

```
print(slope_slow)
```

```
## shot
##   NA
```

```
print(slope_fast)
```

```
## shot
##   NA
```

```
# Complete the code below to build a linear model for approximating *score* as a function of *shot* and *sprint* using the training 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
predicted_scores = predict(model, newdata = hDataTest)

# Print the true and predicted scores for the test data
print(hDataTest$score)
```

```
## [1] 6858 6297 6137 5686 5289
```

```
print(predicted_scores)
```

```
##   Behmer (GDR)  Greiner (USA) Scheider (SWI)   Kytola (FIN) Jeong-Mi (KOR)
##       6549.446       6279.790       5592.081       5613.656       5389.814
```

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

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

```r
# Fit a linear model for approximating *score* as a function of *shot*, *javelin*, and *sp
rint*. Print the model's summary and 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 performa
nce, we can say with 95% confidence that the athlete's score will increase/decrease by an
amount in the interval [?, ?].
model = lm(score ~ shot + javelin + sprint, data = heptathlon)
summary(model)
```

```
##
## Call:
## lm(formula = score ~ shot + javelin + sprint, data = heptathlon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1090.63  -173.25    12.63   203.29   537.00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3349.127   1347.536   2.485  0.02144 *
## shot         249.548     59.669   4.182  0.00042 ***
## javelin       -5.996     22.297  -0.269  0.79061
## sprintslow  -354.060    235.151  -1.506  0.14705
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 345.9 on 21 degrees of freedom
## Multiple R-squared:  0.676,  Adjusted R-squared:  0.6298
## F-statistic: 14.61 on 3 and 21 DF,  p-value: 2.301e-05
```

```r
initial_model = lm(score ~ shot + sprint, data = heptathlon)
improved_accuracy = (summary(model)$r.squared > summary(initial_model)$r.squared)
print(paste("Adding 'javelin' improved model accuracy:", improved_accuracy))
```

```
## [1] "Adding 'javelin' improved model accuracy: TRUE"
```

```r
linear_relationship_javelin = summary(model)$coefficients["javelin", "Pr(>|t|)"] < 0.05
print(paste("The model suggests a possible linear relationship between an athlete's score
and javelin performance:", linear_relationship_javelin))
```

```
## [1] "The model suggests a possible linear relationship between an athlete's score and j
avelin performance: FALSE"
```

```
confidence_interval_shot = confint(model)[2, ]
print(paste("95% Confidence Interval for the effect of shot on score:", confidence_interva
l_shot))
```

```
## [1] "95% Confidence Interval for the effect of shot on score: 125.459919224586"
## [2] "95% Confidence Interval for the effect of shot on score: 373.635209841273"
```

```
# Fit a linear model for approximating *score* as a function of *highjump*, and *sprint*.
Print the model's summary and answer 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(score ~ highjump + sprint, data = heptathlon)
summary(model)
```

```
##
## Call:
## lm(formula = score ~ highjump + sprint, data = heptathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -476.12 -162.88  -29.12  146.92  502.33
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2030.8     1175.5  -1.728   0.0981 .
## highjump      4873.2      646.0   7.544 1.54e-07 ***
## sprintslow    -703.3      123.3  -5.702 9.81e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 241.9 on 22 degrees of freedom
## Multiple R-squared:  0.8341, Adjusted R-squared:  0.819
## F-statistic: 55.29 on 2 and 22 DF,  p-value: 2.625e-09
```

```
r_squared = summary(model)$r.squared
print(paste("R-squared value:", round(r_squared, 4)))
```

```
## [1] "R-squared value: 0.8341"
```

```
p_values = summary(model)$coefficients[, "Pr(>|t|)"]
insignificant_features = names(p_values[p_values > 0.1])
print(paste("Insignificant features:", paste(insignificant_features, collapse = ", ")))
```

```
## [1] "Insignificant features: "
```

The R square value has increased significantly now and there are no insignificant features