AML 5251 | Advanced Applications of Probability and Statistics | Lab Final | Even Semester 2024

Instructions:

1. There are 15 questions;
2. The exam is open book, notes, internet etc. You are welcome to refer to any non-human resource such as ChatGPT, Grok, Bard etc., for answering the questions;
3. However, you must *not* discuss your questions or code with anyone else, inside or outside the class;
4. You should not share the code with anyone else; doing so will result in significant penalties for all involved.
5. By submitting your work, you are implicitly honoring the agreement above;
6. You might be called for a one-on-one during the exam after reviewing your submission to explain your code and answer additional questions. Failure to justify your code and answers will result in significant points docked from your exam score.
7. After finishing the exam, delete all codes related to the exam from the computer you are working on.

Upload the following two files by clicking here

1. completed code clearly showing the output cells (.ipynb file) **with the naming convention example**

   AAPS_LabFinal_SudarsanAcharya.ipynb

   and
2. PDF of your completed code clearly showing the output cells (go to file->print->save as PDF choosing Landscape orientation) **with the naming convention example**

   AAPS_LabFinal_SudarsanAcharya.pdf

```
# Install and load packages
library(ggplot2)
library(dplyr)
install.packages('HSAUR')
library(HSAUR)
```

```
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


Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Loading required package: tools
```

```
# 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 ...
```

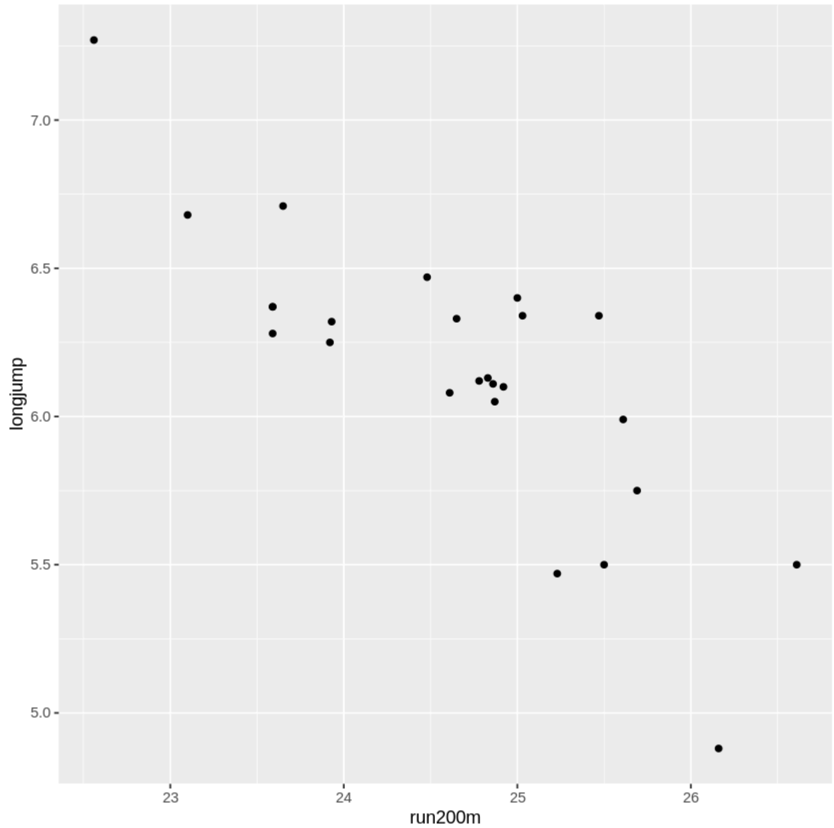
```
# Introduce a new column called sprint highlighting slow and fast sprinters
heptathlon = heptathlon %>% mutate(sprint = ifelse(run200m <= 25 & run800m <= 129, 'fast', 'slow'))

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


# Print the first few rows of the dataframe
head(heptathlon)
```

A data.frame: 6 × 9

	hurdles	highjump	shot	run200m	longjump	javelin	run800m	score	sprint
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<fct>
Joyner-Kersee (USA)	12.69	1.86	15.80	22.56	7.27	45.66	128.51	7291	fast
John (GDR)	12.85	1.80	16.23	23.65	6.71	42.56	126.12	6897	fast
Behmer (GDR)	13.20	1.83	14.20	23.10	6.68	44.54	124.20	6858	fast
Sablovskaite (URS)	13.61	1.80	15.23	23.92	6.25	42.78	132.24	6540	slow
Choubenkova (URS)	13.51	1.74	14.76	23.93	6.32	47.46	127.90	6540	fast
Schulz (GDR)	13.75	1.83	13.50	24.65	6.33	42.82	125.79	6411	fast

Question-1: Make a scatter plot between *run200m* (x-axis) and *longjump* (y-axis). What do you observe from this plot?

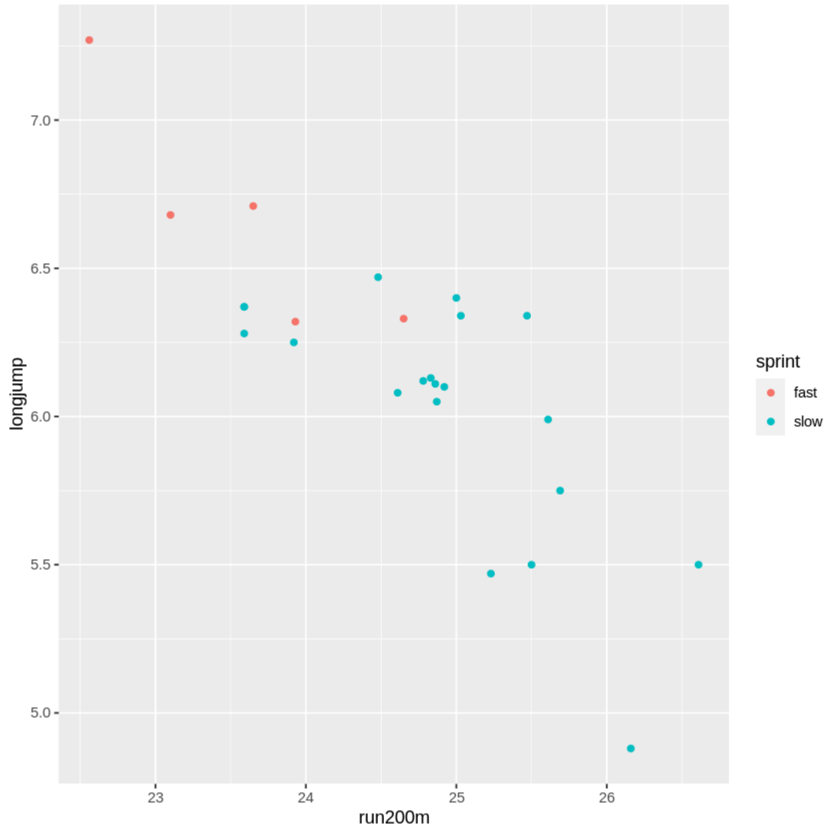
```
p1 = ggplot(data = heptathlon, aes(x = run200m, y = longjump)) +
  geom_point()
p1
```



From the graph we can tell that there is a negative correlation between longjump and run200m i.e if the value of run200m increases then longjump value decreases

Question-2: Make a scatter plot between *run200m* (x-axis) and *longjump* (y-axis) with the data points color-coded using *sprint*. What do you observe from this plot?

```
p2 = ggplot(data = heptathlon, aes(x = run200m, y = longjump, color = sprint)) +
  geom_point()
p2
```



From the above graph we can tell that the athletes who are fast sprinters generally have higher longjump values than athletes who are slow sprinters and also longjump and run200m are negatively correlated

Question-3: Calculate Pearson's correlation between *run200m* and *longjump*. What do you observe?

```
cor(heptathlon$run200m, heptathlon$longjump, method = 'pearson')
```

-0.817205299701264

Pearson's correlation value of -0.817 indicates a strong negative correlation between run200m and longjump

Question-4: Select data frame without *sprint* and *score* columns.

```
hData = heptathlon %>% select(-c(sprint,score))
```

```
hData
```

A data.frame: 25 × 7

	hurdles	highjump	shot	run200m	longjump	javelin	run800m
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Joyner-Kersee (USA)	12.69	1.86	15.80	22.56	7.27	45.66	128.51
John (GDR)	12.85	1.80	16.23	23.65	6.71	42.56	126.12
Behmer (GDR)	13.20	1.83	14.20	23.10	6.68	44.54	124.20
Sablovskaite (URS)	13.61	1.80	15.23	23.92	6.25	42.78	132.24
Choubenkova (URS)	13.51	1.74	14.76	23.93	6.32	47.46	127.90
Schulz (GDR)	13.75	1.83	13.50	24.65	6.33	42.82	125.79
Fleming (AUS)	13.38	1.80	12.88	23.59	6.37	40.28	132.54
Greiner (USA)	13.55	1.80	14.13	24.48	6.47	38.00	133.65
Lajbnerova (CZE)	13.63	1.83	14.28	24.86	6.11	42.20	136.05
Bouraga (URS)	13.25	1.77	12.62	23.59	6.28	39.06	134.74
Wijnsma (HOL)	13.75	1.86	13.01	25.03	6.34	37.86	131.49
Dimitrova (BUL)	13.24	1.80	12.88	23.59	6.37	40.28	132.54
Scheider (SWI)	13.85	1.86	11.58	24.87	6.05	47.50	134.93
Braun (FRG)	13.71	1.83	13.16	24.78	6.12	44.58	142.82
Ruotsalainen (FIN)	13.79	1.80	12.32	24.61	6.08	45.44	137.06
Yuping (CHN)	13.93	1.86	14.21	25.00	6.40	38.60	146.67
Hagger (GB)	13.47	1.80	12.75	25.47	6.34	35.76	138.48
Brown (USA)	14.07	1.83	12.69	24.83	6.13	44.34	146.43
Mulliner (GB)	14.39	1.71	12.68	24.92	6.10	37.76	138.02
Hautenauve (BEL)	14.04	1.77	11.81	25.61	5.99	35.68	133.90

Kytola (FIN)	14.31	1.77	11.66	25.69	5.75	39.48	133.35
Geremias (BRA)	14.23	1.71	12.95	25.50	5.50	39.64	144.02
Hui-Ing (TAI)	14.85	1.68	10.00	25.23	5.47	39.14	137.30
Jeong-Mi (KOR)	14.53	1.71	10.83	26.61	5.50	39.26	139.17
Launa (PNG)	16.42	1.50	11.78	26.16	4.88	46.38	163.43

Question 5: Using the code in the cell below, answer the following questions:

1. Which principal component assigns the greatest weight (in magnitude) to *run200m*?
2. Which principal component assigns the greatest weight (in magnitude) to *longjump*?
3. *True/false*: the 2nd principal component score for a sample assigns a maximum weight to *javelin*.
4. The 1st principal component assigns the least weight (in magnitude) to which feature?

Does using the correlation matrix change your answers to the above questions? Which one will you finally use for dimension reduction using PCA: the covariance or the correlation matrix?

```
# Calculate eigenvalues & eigenvectors of sample covariance matrix
e = eigen(cor(hData))

# Eigenvectors of the sample covariance matrix
u =e$vectors

# Eigenvalues of the sample covariance matrix
lambda = e$values


print(lambda)
```

```
[1] 69.967253281 12.895102688  1.920157728  0.343059843  0.104857334
[6]  0.021644924  0.001105536
```

```
head(hData)
```

A data.frame: 6 × 7

	hurdles	highjump	shot	run200m	longjump	javelin	run800m
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Joyner-Kersee (USA)	12.69	1.86	15.80	22.56	7.27	45.66	128.51
John (GDR)	12.85	1.80	16.23	23.65	6.71	42.56	126.12
Behmer (GDR)	13.20	1.83	14.20	23.10	6.68	44.54	124.20
Sablovskaite (URS)	13.61	1.80	15.23	23.92	6.25	42.78	132.24
Choubenkova (URS)	13.51	1.74	14.76	23.93	6.32	47.46	127.90
Schulz (GDR)	13.75	1.83	13.50	24.65	6.33	42.82	125.79

```
print(u)
```

```
           [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
[1,]  0.4528710 -0.15792058  0.04514996 -0.02653873 -0.09494792  0.78334101
[2,] -0.3771992  0.24807386 -0.36777902  0.67999172 -0.01879888  0.09939981
[3,] -0.3630725 -0.28940743  0.67618919  0.12431725 -0.51165201 -0.05085983
[4,]  0.4078950  0.26038545 -0.08359211  0.36106580 -0.64983404 -0.02495639
[5,] -0.4562318  0.05587394  0.13931653  0.11129249  0.18429810  0.59020972
[6,] -0.0754090 -0.84169212 -0.47156016  0.12079924 -0.13510669 -0.02724076
[7,]  0.3749594 -0.22448984  0.39585671  0.60341130  0.50432116 -0.15555520
            [,7]
[1,]  0.38024707
[2,]  0.43393114
[3,]  0.21762491
[4,] -0.45338483
[5,] -0.61206388
[6,] -0.17294667
[7,] -0.09830963
```

1. Which principal component assigns the greatest weight (in magnitude) to *run200m*?

Principal Component 4 assigns the greatest weight (in magnitude) to run200m

```
max(0.072967545, 0.1012004268, 0.31005700 ,0.81585220,-0.46178680 , 0.082486244 ,-0.051312974)
```

0.8158522

2. Which principal component assigns the greatest weight (in magnitude) to *longjump*?

Principal Component 6 assigns the greatest weight (in magnitude) to *longjump*

```
max(-0.040369299 ,-0.0148845034 ,-0.18494319 ,-0.20419828, -0.31899315,0.894592570 ,-0.142110352)
```

0.89459257

3. *True/false*: the 2nd principal component score for a sample assigns a maximum weight to *javelin*.

True the 2nd principal component score for a sample assigns a maximum weight to *javelin* which is 0.9852954510

4. The 1st principal component assigns the least weight (in magnitude) to which feature?

The 1st principal component assigns the least weight (in magnitude) to *highjump* feature which is 0.005569781

Does using the correlation matrix change your answers to the above questions? Which one will you finally use for dimension reduction using PCA: the covariance or the correlation matrix

yes as correlation matrix is scaled the values will increase covariance matrix

```
X = scale(hData)

e = eigen(cov(X))
V = e$vectors
lambda = e$values
colnames(hData)
print(V)
print(lambda)
```

'hurdles' · 'highjump' · 'shot' · 'run200m' · 'longjump' · 'javelin' · 'run800m'

```
           [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
[1,]  0.4528710 -0.15792058  0.04514996 -0.02653873 -0.09494792  0.78334101
[2,] -0.3771992  0.24807386 -0.36777902  0.67999172 -0.01879888  0.09939981
[3,] -0.3630725 -0.28940743  0.67618919  0.12431725 -0.51165201 -0.05085983
[4,]  0.4078950  0.26038545 -0.08359211  0.36106580 -0.64983404 -0.02495639
[5,] -0.4562318  0.05587394  0.13931653  0.11129249  0.18429810  0.59020972
[6,] -0.0754090 -0.84169212 -0.47156016  0.12079924 -0.13510669 -0.02724076
[7,]  0.3749594 -0.22448984  0.39585671  0.60341130  0.50432116 -0.15555520
            [,7]
[1,]  0.38024707
[2,]  0.43393114
[3,]  0.21762491
[4,] -0.45338483
[5,] -0.61206388
[6,] -0.17294667
[7,] -0.09830963
[1] 4.46027516 1.19432056 0.52101413 0.45716683 0.24526674 0.07295558 0.04900101
```

Question-6: Explain the output of the cell below?

```
# Extract data matrix from data frame
X = as.matrix(hData)

print(X %*% u[, 1])
```

```
                             [,1]
Joyner-Kersee (USA) 128.6514
John (GDR)          126.3423
```

```
Behmer (GDR)         124.5962
Sablovskaite (URS)   132.5776
Choubenkova (URS)    128.3359
Schulz (GDR)         126.3804
Fleming (AUS)        132.9964
Greiner (USA)        134.0565
Lajbnerova (CZE)     136.4989
Bouraga (URS)        135.1834
Wijnsma (HOL)        132.0612
Dimitrova (BUL)      132.9867
Scheider (SWI)       135.6531
Braun (FRG)          143.3104
Ruotsalainen (FIN)   137.6684
Yuping (CHN)         147.0239
Hagger (GB)          139.0075
Brown (USA)          146.9512
Mulliner (GB)        138.6044
Hautenauve (BEL)     134.6055
Kytola (FIN)         134.1319
Geremias (BRA)       144.5973
Hui-Ing (TAI)        138.1891
Jeong-Mi (KOR)       140.0555
Launa (PNG)          164.1953
```

The output represents the contribution of each sample to the variation captured by the 1st PC Score

Question-7: Explain the output of the cell below?

```
print(var(X %*% u[, 2]))
```

```
        [,1]
[1,] 12.8951
```

It is the total variance captured by the 2nd Principal Component

Question-8: How many minimum principal components are needed to explain more than 90% of the variance in the data? In one line, explain how you could use the corresponding principal component scores (projected values) to get a final score for each athlete so that they can be ranked.

```
cumsum(lambda)
```

69.9672532806531 · 82.862355968615 · 84.7825136966759 · 85.1255735392405 · 85.2304308736865 · 85.2520757974337 · 85.2531813333333

```
sum(lambda)
```

85.2531813333333

```
variance <- cumsum(lambda) / sum(lambda)
variance
```

0.820699617144921 · 0.971956174217472 · 0.994479178028358 · 0.998503190237631 · 0.999733142396671 · 0.999987032320879 · 1

```
num_components_90 <- which.max(variance > 0.9)
num_components_90
```

2

How many minimum principal components are needed to explain more than 90% of the variance in the data?

2 Principal Components

In one line, explain how you could use the corresponding principal component scores (projected values) to get a final score for each athlete so that they can be ranked.

Add their scores from each selected principal component. This combined score reflects the athlete's overall performance across different aspects captured by those principal components. Based on this we can rank them

```
scores <- X %*% u[, 1:num_components_90]

final_scores <- rowSums(scores)


ranked_athletes <- order(final_scores, decreasing = TRUE)
ranked_athletes
```

25 · 16 · 22 · 17 · 18 · 24 · 19 · 23 · 20 · 14 · 10 · 8 · 21 · 11 · 9 · 7 · 12 · 15 · 4 · 13 · 6 · 2 · 1 · 5 · 3

Question 9: how many levels does the categorical variable *sprint* have? What is the reference level?

```
contrasts(heptathlon$sprint)
```

A matrix: 2 × 1 of type dbl

	slow
fast	0
slow	1

how many levels does the categorical variable *sprint* have? What is the reference level?

2 levels with reference level fast

Question 10: fit a linear model for approximating *score* as a function of *shot* and *sprint*. Print the model's summary. How accurate is the model?

```
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
```

The model is 64.54% accurate

Question 11: fit a linear model for approximating *score* as a function of *shot*, *javelin*, and *sprint*. 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 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 + javelin)
summary(model)
```

```
Call:
lm(formula = score ~ shot + sprint + javelin, 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 ***
sprintslow  -354.060    235.151  -1.506  0.14705    
javelin       -5.996     22.297  -0.269  0.79061    
---
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
```

```
249.548-2*59.669
```

130.21

```
249.548+2*59.669
```

368.886

1. Did the addition of the new predictor *javelin* improve the model accuracy?

No the model accuracy decreased

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.

False there is p value which is very much greater than 5% or even 10% threshold therefore there is no possible linear relationship between javelin and athlete's score

3. For a 1 metre increase in shot put throw and with the same javelin and sprint performance, we can say with 95% confidence that the athlete's score will increase/decrease by an amount in the interval [?, ?].

For a 1 metre increase in shot put throw and with the same javelin and sprint performance, we can say with 95% confidence that the athlete's score will increase/decrease by an amount in the interval [130.21, 368.886]

Question 12: 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(data = heptathlon, score ~ highjump + sprint )
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
```

1. How accurate is this model?

81.9% accurate

2. Considering a p-value of 10% as cutoff, are there any insignificant features?

NO there are no insignificant features

Question 13: Using the model built above, extract the slope and intercept for estimating the *score* of *slow* and *fast* athletes. It would be helpful to start with the regression equation $\hat{y}^{(i)} = \hat{\beta}_0 + \hat{\beta}_1 x_1^{(i)} + \hat{\beta}_2 x_2^{(i)}$, and then write two separate equations for *slow* and *fast* athletes.

```
coef = coef(model)
```

(Intercept): -2030.827929374 highjump: 4873.19422150883 sprintslow: -703.255216693418

```
coef <- coef(model)


intercept_slow <- coef["(Intercept)"] + coef["sprintslow"]
slope_slow <- coef["highjump"]


intercept_fast <- coef["(Intercept)"]
slope_fast <- coef["highjump"]


print(paste("Intercept for slow athletes:", intercept_slow))
print(paste("Slope for slow athletes:", slope_slow))
print(paste("Intercept for fast athletes:", intercept_fast))
print(paste("Slope for fast athletes:", slope_fast))
```

```
[1] "Intercept for slow athletes: -2714.515940448"
[1] "Slope for slow athletes: 3889.38458022362"
[1] "Intercept for fast athletes: -2303.48561353747"
[1] "Slope for fast athletes: 3889.38458022362"
```

Question 14: fit a linear model for approximating *score* as a function of *shot*, *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?

89.59%

there is no insignificant

```
model = lm(data = heptathlon, score ~ highjump + sprint + shot )
```