**Question 1.**

Throughout this question, when comparing matrices, please use numpy.allclose() in your answers in an appropriate way.

Consider the matrix given below.

The Python code below constructs matrices and so that .

import scipy.linalg as la

A, B = la.polar(M)

1. Write Python code to enter the matrix as a numpy array, print out , print out the corresponding matrices and , and check that we do have .

|  |
| --- |
| Code:  **import** **numpy** **as** **np**  **import** **scipy.linalg** **as** **la**  M = np.array([[**3**,**8**,**2**],[**2**,**5**,**7**],[**1**,**4**,**6**]])  **print**('M',M)  A,B=la.polar(M)  **print**('A',A)  **print**('B',B)  **print**('A@B',A**@B**)  **print**(np.allclose(A**@B**,M))  Output:  M [[3 8 2]  [2 5 7]  [1 4 6]]  A [[ 0.30193284 0.91751954 -0.25883288]  [ 0.6773784 -0.01543097 0.7354729 ]  [-0.67081671 0.39739122 0.62616703]]  B [[1.58973862 3.11908789 1.3206142 ]  [3.11908789 8.85256633 4.11136962]  [1.3206142 4.11136962 8.38764675]]  A@B [[3. 8. 2.]  [2. 5. 7.]  [1. 4. 6.]]  True |

1. What kinds of matrices are and (e.g. symmetric, diagonal, etc)? What do you notice about ? *Include both Python code and output in your answer. Use numpy to check your answers.*

|  |
| --- |
| **Code:**  **print**(np.allclose(A**@A.T**,np.identity(**3**)))  **print**('A is orthogonal , A@A.T = I')  **print**(np.isclose(B,B.T))  **print**('B is symmetric, B = B.T')  Ainv=np.linalg.inv(A)  **print**('Ainv',Ainv)  **print**('A+Ainv.T:',A+Ainv.T)  **print**(**2**\*A)  **print**(np.allclose(**2**\*A,(A+Ainv.T)))  Output:  True  A is orthogonal , A@A.T = I  [[ True True True]  [ True True True]  [ True True True]]  B is symmetric, B = B.T  Ainv [[ 0.30193284 0.6773784 -0.67081671]  [ 0.91751954 -0.01543097 0.39739122]  [-0.25883288 0.7354729 0.62616703]]  A+Ainv.T: [[ 0.60386569 1.83503907 -0.51766576]  [ 1.3547568 -0.03086194 1.4709458 ]  [-1.34163343 0.79478245 1.25233406]]  [[ 0.60386569 1.83503907 -0.51766576]  [ 1.3547568 -0.03086194 1.4709458 ]  [-1.34163343 0.79478245 1.25233406]]  True |

1. The Python code below finds the eigenvalues and eigenvectors of , where is given above.

V, P = np.linalg.eig(M.T@M)

Write Python code to construct (and print out) a diagonal matrix where the elements on the diagonal of are the square roots of the eigenvalues of . Investigate how is related to the and from part (a).

|  |
| --- |
| Code:  V,P= np.linalg.eig(M.T**@M**)  **print**(V)  **print**(P)  Eig\_V = np.sqrt(V)  C = np.diag(Eig\_V)  **print**('C:' , C)  Pinv=np.linalg.inv(P)  **print**('P@C@Pinv:',P**@C@Pinv**)  **print**(np.isclose(B,P**@C@Pinv**))  Output:  [1.84648905e+02 1.83266888e-01 2.31678280e+01]  [[ 0.25868492 0.93220675 -0.25312584]  [ 0.72475028 -0.36054697 -0.58714812]  [ 0.63860719 0.03156666 0.76888517]]  C: [[13.58855787 0. 0. ]  [ 0. 0.42809682 0. ]  [ 0. 0. 4.813297 ]]  P@C@Pinv: [[1.58973862 3.11908789 1.3206142 ]  [3.11908789 8.85256633 4.11136962]  [1.3206142 4.11136962 8.38764675]]  [[ True True True]  [ True True True]  [ True True True]] |

**Question 2.**

The *Beta* distribution is commonly used in Bayesian data analysis. A Beta distribution is governed by two parameters: and . The Python code below plots the probability density function (pdf) of the particular Beta distribution with the given values of and .

import scipy.stats as stats

import numpy as np

import matplotlib.pyplot as plt

a = 2

b = 5

x = np.arange(0,1.01,0.01)

y = stats.beta.pdf(x, a, b)

plt.figure()

plt.plot(x,y,'b-')

plt.grid()

plt.show()

1. The *mean* of a Beta distribution is exactly . Calculate the mean, and the upper quartile (75% percentile) of the particular Beta distribution above and add them to the plot using vertical dashed lines. *Include both Python code and output in your answer.*

|  |
| --- |
| Output plot of pdf:    Code:  m = a/(a+b)  UQ = np.percentile(y,**75**)  x = np.arange(**0**,**1.01**,**0.01**)  y = stats.beta.pdf(x, m, UQ)  plt.figure()  plt.vlines(x,y,'b--')  plt.grid()  plt.show()  Output: |

1. It is claimed that the following Python code generates a sample from a Beta distribution.

sample = []

for i in range(100000):

A = np.random.random(a+b-1)

A.sort()

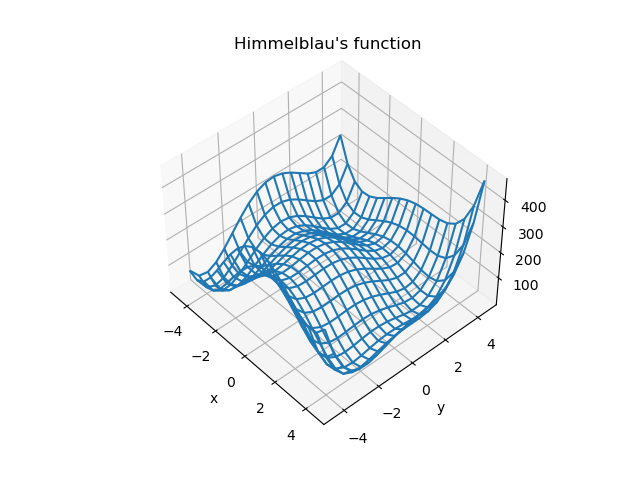
sample.append(A[a-1])

Plot a histogram (using 50 bins) of the sample produced using the Python code above. Calculate the mean and upper quartile from this sample and compare to the equivalent values from part (a). Does the sample appear to be random values from the particular Beta distribution from part (a)? *Justify your answer.* *Include both Python code, histogram and output in your answer.*

|  |
| --- |
| Code to plot histogram:  **print**(sample)  plt.figure()  plt.hist(sample,bins=**50**)  plt.show()  output:    Does the sample appear to be random values from the particular Beta distribution from part (a)?  Answer: the sample values from this code and random values from part (a) looks similar. |

**Question 3.**

Consider *Himmelblau’s* function and the corresponding 3D wireframe plot given below.



1. Briefly explain what is mean by a “contour” on a contour plot, i.e., what does each individual contour represent?

|  |
| --- |
| Contour means an outline.( an outline especially of a curving or irregular figure). Contour joins points of equal elevation (height) above a given level.  A contour plot is a graphical technique for representing a 3-dimensional surface by plotting constant z slices, called contours, on a 2-dimensional format. That is, given a value for z, lines are drawn for connecting the (x,y) coordinates where that z value occurs.  Contour lines allow a vertical dimension to be added to a map and represent elevations above sea level. Since each individual contour line connects points of equal elevation, then following that line in the real world means that you are staying at the same elevation while walking along that imaginary line. |

1. Write Python code to draw a contour plot of Himmelblau’s function. Carefully select the contours (levels) to clearly highlight the valley bottoms and hill tops. Show that there is at least one contour that crosses itself. *Include both Python code and contour plot in your answer.*

|  |
| --- |
| Code:  **import** **numpy** **as** **np**  **import** **matplotlib.pyplot** **as** **plt**  x = np.arange(-**7**,**7**,**0.01**)  y = np.arange(-**5**,**5**,**0.01**)  X,Y = np.meshgrid(x,y)  Z = ((X\*\***2**+Y-**11**)\*\***2**+(X+Y\*\***2**-**8**)\*\***2**)  levels = [**50**,**100**,**150**,**200**,**250**,**300**,**350**,**400**,**450**,**500**]  plt.figure()  plt.contour(X,Y,Z,levels)  plt.title('Himmelblau’s function')  plt.axis('equal')  plt.grid()  plt.xlabel('x')  plt.ylabel('y')  plt.show()    Output:    Plot to show Contour crossing itself:  **import** **numpy** **as** **np**  **import** **matplotlib.pyplot** **as** **plt**  x = np.arange(-**7**,**7**,**0.01**)  y = np.arange(-**5**,**5**,**0.01**)  X,Y = np.meshgrid(x,y)  Z = ((X\*\***2**+Y-**11**)\*\***2**+(X+Y\*\***2**-**8**)\*\***2**)  levels = [**50**,**55**,**60**,**65**,**70**,**75**,**80**,**85**,**100**,**150**,**200**,**250**,**300**,**350**,**400**,**450**,**500**]  plt.figure()  plt.contour(X,Y,Z,levels)  plt.title('Himmelblau’s function')  plt.axis('equal')  plt.grid()  plt.xlabel('x')  plt.ylabel('y')  plt.show()  plot: |

**Question 4.**

Consider the dataset in the file “landcover.csv” (provided along with these questions on Moodle) on the area of land (the variable *Area*) covered by different land cover classes (the variable *Class*) in Brazil and Canada in the years 2008 and 2018. The data comes from the Food and Agriculture Organization of the United Nations (FAO) statistics <http://www.fao.org/faostat/>.

*In RStudio, make sure you go to the Session menu, select Set Working Directory and then Source File Location. Save the “landcover.csv” file to the same folder as where your R code is saved.*

library(tidyverse)

land = read\_csv('landcover.csv')

1. Write R code to find the number of rows and the names of the columns in this dataset. *Include only your R code in your answer.*

|  |
| --- |
| Code:  library(tidyverse)  lc = read\_csv('landcover.csv')  nrow(lc)  colnames(lc)  output: |

1. Write R code to find the unique values of land cover classes in this dataset. *Include both R code and output in your answer.*

|  |
| --- |
| Code:  unique(lc[c("Class")])  Output: |

1. Consider only the year 2018. Use R to find the total area of land of each *Country*. Also write a dplyr pipe to construct a summary table giving the total area for each land cover class, given in decreasing order of area. *Include both R code and output in your answer.*

|  |
| --- |
| Code:  lc\_sort1 = lc %>%  filter(Year==**2018**) %>%  group\_by(Country) %>%  summarise(total\_area = sum(Area))  lc\_sort1  lc\_sort = lc %>%  filter(Year==**2018**) %>%  group\_by(Class) %>%  summarise(total\_area=sum(Area)) %>%  arrange(total\_area)  lc\_sort  output: |

**Question 5.**

Consider the dataset “nutrition.csv” (provided along with these questions on Moodle) which gives the sale *price* and nutritional data for various menu items from a well-known fast food restaurant in 2010.

*In RStudio, make sure you go to the Session menu, select Set Working Directory and then Source File Location. Save the “nutrition.csv” file to the same folder as where your R code is saved.*

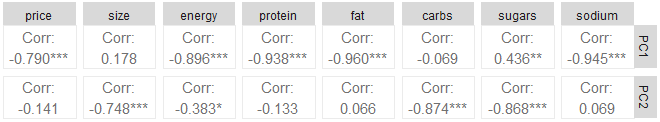
library(tidyverse)

nutrition = read\_csv('nutrition.csv')

library(GGally)

nutrition %>% select(-name) %>% ggpairs()

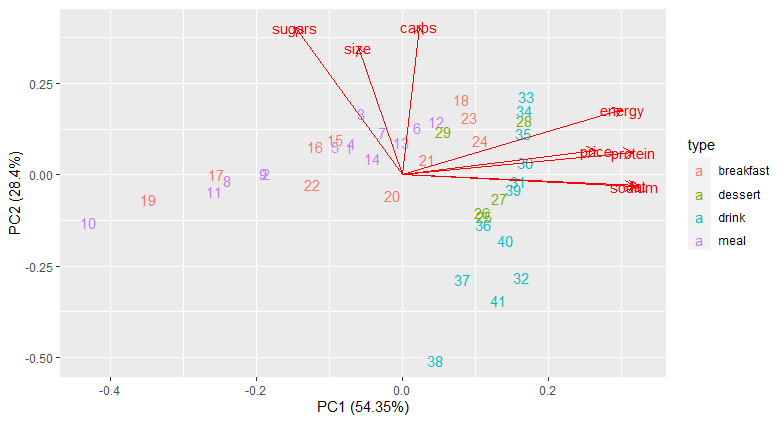
1. Scaling the dataset and applying PCA gives the partial scatter matrix below showing the correlations between the first two principal components (PC1 and PC2) and the eight original quantitative variables.



Use R to produce a PCA loadings plot showing PC1 and PC2. By considering the loadings plot, the variance explained by each principal component, and the partial scatter matrix above, give one possible interpretation of each of PC1 and PC2. Justify whether PC1 and PC2 alone are sufficient. *Include your R code and output in your answer.*

|  |
| --- |
| nutrition\_sort = select(nutrition,-name,-type)  nutrition\_sort    pca\_final = prcomp(nutrition\_sort,scale.=TRUE)  pca\_final    autoplot(pca\_final,  label=TRUE,label.size=**3**,shape=FALSE,  loadings=TRUE,loadings.label=TRUE,  data=nutrition,colour='type')    var\_explained = **100**\*((pca\_final$sd)^**2**)/(sum((pca\_final$sd)^**2**))  var\_explained  cumsum(var\_explained)  ggplot(NULL,aes(x=**1**:**8**,y=var\_explained)) +  geom\_col() +  ggtitle('scree plot') +  xlab('Principal Component (PC)') +  ylab('Percentage Variance Explained')      loadings = **as**.data.frame(pca\_final$rotation[,**1**:**2**])  loadings$symbol = row.names(loadings)  loadings = gather(loadings,key='component',value = 'X1',-symbol)  ggplot(loadings,aes(x=symbol,y=X1))+  geom\_bar(stat = 'identity') +  facet\_grid(component~.) +  ggtitle('Loadings for PC1-PC2')    PC1 (54.34%) + PC2 (28.39%) gives 82.73% of variance.  #we can see that 82% of the variance is measured by the first two PC’s.  #    **PCA**   * Principal Component Analysis (PCA) is a useful technique for exploratory data analysis, allowing you to better visualize the variation present in a dataset with many variables. It is particularly helpful in the case of "wide" datasets, where you have many variables for each sample. * PCA allows you to see the overall "shape" of the data, identifying which samples are similar to one another and which are very different. This can enable us to identify groups of samples that are similar and work out which variables make one group different from another. * Using Prcomp() method,we can find the standard deviation and correlation coefficient for all the five measurements of all the died/survived sparrows. * Here PC1 (54.34997825%) + PC2 (28.39725863%) gives 82.73% of variance. we can see that 83% of the variance is measured by the first two PC’s. so PC1 and PC2 are sufficient   Also produced Biplot, Scree plot and Loading plot.   * In the Biplot between PC1 and PC2 ,the axes are seen as arrows originating from the center point. Here, we see that the variables X1,X2,X3,X4 and X5 all contribute to PC1, with higher values in those variables moving the samples to the right on this plot.The x-axis displays the PC1 and the y-axis displays the PC2 with respect to GROUP(died/survival). * In Scree plot, we’ll calculate the percentage of total variance explained by each principal component. The x-axis displays the principal component and the y-axis displays the percentage of total variance explained by each individual principal component. * The first principal component explains **54.34997825%** of the total variation in the dataset. * The second principal component explains **28.39725863 %** of the total variation in the dataset. * The third principal component explains **9.69735637%** of the total variation in the dataset. * The fourth principal component explains **3.67262734%** of the total variation in the dataset. * The fifth principal component explains **1.94312225%** of the total variation in the dataset. * The sixth principal component explains **1.41372446%** of the total variation in the dataset. * The seventh principal component explains **0.51451082%** of the total variation in the dataset. * The eigth principal component explains **0.01142189%** of the total variation in the dataset.   Notice that all of the percentages sum to 100%.   * In loadings plot, it shows the effect of each measurement to construct PCA. Loading means Eigen vector, which means coefficient. Loadings plot means representing coefficient in a plot.PC1 is weighted average of the measurement because all the 8 measurements are positive. For PC1,PC2,PC3,PC4,PC5,PC6,PC7 and PC8 some of the measurements are negative as it explains less percentage if the total variation in the dataset. |

1. Consider the PCA biplot given below. Use R to produce an appropriate dendrogram giving a hierarchical clustering of individual menu items (rows of the dataset). Compare the clusters formed in the dendrogram (using FOUR clusters) with the clusters of individual menu items visible in the PCA biplot (consider both location of clusters and types of menu items, i.e., breakfast, dessert, etc). *Include any R code and dendrogram you produce in your answer.*



|  |
| --- |
| #1 Euclidean Ward:  library(cluster)  BBB = scale(nutrition\_sort)  D = dist(BBB,method="euclidean")  cluster\_results = agnes(D,method="ward")  plot(cluster\_results,which.plots=**2**,  main='cluster the data(Euclidean/ward)')  rect.hclust(cluster\_results, k=**5**, border=**3**)  plot(**as**.dendrogram(cluster\_results),horiz=FALSE)  clusters = cutree(cluster\_results,k=**5**)  dendogram:    Euclidean -Ward : Agglomerative Coefficient D = 0.92  #2 Euclidean Single:  library(cluster)  BBB = scale(nutrition\_sort)  D = dist(BBB,method="euclidean")  cluster\_results = agnes(D,method="single")  plot(cluster\_results,which.plots=**2**,  main='cluster the data(Euclidean/single)')  rect.hclust(cluster\_results, k=**5**, border=**3**)  plot(**as**.dendrogram(cluster\_results),horiz=FALSE)  clusters = cutree(cluster\_results,k=**5**)    Euclidean -Ward : Agglomerative Coefficient D = 0.61  #3 Manhattan Ward:  library(cluster)  BBB = scale(nutrition\_sort)  D = dist(BBB,method="manhattan")  cluster\_results = agnes(D,method="ward")  plot(cluster\_results,which.plots=**2**,  main='cluster the data(Manhattan/ward')  rect.hclust(cluster\_results, k=**5**, border=**3**)  plot(**as**.dendrogram(cluster\_results),horiz=FALSE)  clusters = cutree(cluster\_results,k=**5**)    Manhattan -Ward : Agglomerative Coefficient D = 0.94  #4 Manhattan Single:  library(cluster)  BBB = scale(nutrition\_sort)  D = dist(BBB,method="manhattan")  cluster\_results = agnes(D,method="single")  plot(cluster\_results,which.plots=**2**,  main='cluster the data(Manhattan/single')  rect.hclust(cluster\_results, k=**5**, border=**3**)  plot(**as**.dendrogram(cluster\_results),horiz=FALSE)  clusters = cutree(cluster\_results,k=**5**)    Manhattan -Single : Agglomerative Coefficient D = 0.6  Implemented the two-distance metrics (Euclidean and Manhattan) using Two hierarchical methods (single, ward) and found Agglomerative Coefficient D. Both results from Euclidean and Manhattan distance metrics looks nearly similar, but Euclidean Distance value is little less when compared to Manhattan Distance. |

**Question 6.**

The dataset in the file “penguins.csv” (provided along with these questions on Moodle) contains data about penguins in Antarctica.

*In RStudio, make sure you go to the Session menu, select Set Working Directory and then Source File Location. Save the “penguins.csv” file to the same folder as where your R code is saved.*

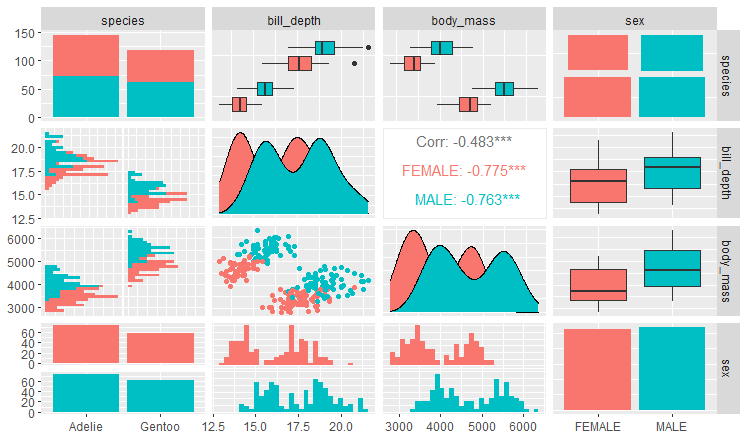
library(tidyverse)

penguins = read\_csv('penguins.csv')

library(GGally)

ggpairs(penguins, aes(colour=sex))

1. The scatter matrix below is produced by the R code given above. Use only the information in the scatter matrix to comment (as comprehensively as possible) on the relationship between *bill\_depth* (measured in mm) and *body\_mass* (measured in g).



|  |
| --- |
| Relation between bill\_depth and body\_mass   * Blue colour represents male and red color represents female. By seeing scatter matrix which is between bill\_depth and body\_mass , we can say that the male is increasing ,female is decreasing and viceversa. * The correlation between bill\_depth and body\_mass is -0.483.A negative correlation **describes the extent to which two variables move in opposite directions**. For example, for two variables, X and Y, an increase in X is associated with a decrease in Y. A negative correlation coefficient is also referred to as an inverse correlation. |

1. Suppose we wish to predict the *body\_mass* of penguins from their *bill\_depth* only. Write R code to fit the linear model “body\_mass~bill\_depth” to the dataset and construct an appropriate scatterplot including the line of best fit. Write down the equation of the fitted model. *Include R code and scatterplot in your answer.*

|  |
| --- |
| Code to fit the linear model “body\_mass~bill\_depth”:  library(tidyverse)  library(GGally)  library(stats)  library(olsrr)  library(ggfortify)  model = lm(body\_mass~bill\_depth,data=penguins)  scatter plot between body\_mass and bill\_depth:  ggplot(penguins,aes(x=bill\_depth,y=body\_mass))+  geom\_point(aes(colour=sex))    Equation of the fitted model:  body mass=intercept + slope\* bill \_depth  body mass=7729.34 - 201.91\* bill \_depth |

1. Assess whether the linear model “body\_mass~bill\_depth” is a good fit to the dataset by considering the corresponding Residuals vs Fitted and Normal Q-Q diagnostic plots produced by the R code given below.

library(ggfortify)

autoplot(model, data=penguins, colour='species')

|  |
| --- |
| library(tidyverse)  library(GGally)  library(stats)  library(olsrr)  library(ggfortify)  model = lm(body\_mass~bill\_depth,data=penguins)  summary(model)  AIC(model)  autoplot(model)  # AIC = 4255.108 , multiple R\_square = 0.23, significant=\*\*\* |

1. Suppose we wish to predict the *body\_mass* of penguins using both *bill\_depth* and *species*. Write R code to fit the linear model “body\_mass~bill\_depth+species” to the dataset and write down the equation of the fitted model.

|  |
| --- |
| code to fit the linear model “body\_mass~bill\_depth+species”:  modelB =lm(body\_mass~bill\_depth+species,data=penguins)  summary(modelB)  AIC(modelB)  autoplot(modelB)      Equation of the fitted model:  body\_mass=-1249.97 - 270.13\* bill \_depth+2297.31\*species |

1. Consider the linear models from the previous parts of this question, i.e., model A (“body\_mass~bill\_depth”) and model B (“body\_mass~bill\_depth+species”). Give a comparison between these two fitted models. Which of these models might you consider to be a “better” model for predicting *body\_mass* and why?

|  |
| --- |
| library(tidyverse)  library(GGally)  library(stats)  library(olsrr)  library(ggfortify)  model\_A = lm(body\_mass~bill\_depth,data=penguins)  summary(model\_A)  AIC(model\_A)    model\_B = lm(body\_mass~bill\_depth+species,data=penguins)  summary(model\_B)  AIC(model\_B)    ggplot(penguins,aes(x=model\_A$residuals, y=model\_B$residuals,colour=sex)) +  geom\_point()    cor(model\_A$residuals, model\_B$residuals)  Output: 0.5027719  ggplot(penguins,aes(x=body\_mass))+  geom\_point(aes(y=model\_A$residual),shape=2,size=5)+  geom\_point(aes(y=model\_B$residuals),shape=16,size=5)+  geom\_segment(aes(x=body\_mass,xend=body\_mass,y=model\_A$residuals,yend=model\_B$residuals))    library(olsrr)  model= lm(body\_mass~.,data=penguins)  ols\_step\_best\_subset(model)    Code to find “better” model for predicting *body\_mass* and why?  To know the model is best fit for my data, there are some most common metrics to look at while selecting the model.   * R\_squared – Higher the better * AIC – Lower the better   AIC value of model\_A = 4255.108 and AIC value of model\_B is 3892.67  R\_squared for model\_A = 0.23 and R\_squared for model\_B = 0.8  So with the above calculated values, model\_B is the better model for predicting body\_mass. |