Week10_Assignment1_KoppulaVeera

Veera Koppula

August 09 2021

##These assignments are here to provide you with an introduction to the "Data Science" use for these tools. This is your future. It may seem confusing and weird right now but it hopefully seems far less so than earlier in the semester. Attempt these homework assignments. You will not be graded on your answer but on your approach. This should be a, "Where am I on learning this stuff" check. If you can't get it done, please explain why.

```
## Loading required package: splines

## Loading required package: RcmdrMisc

## Loading required package: car

## Loading required package: carData

## Loading required package: sandwich

## Loading required package: effects

## lattice theme set by effectsTheme()
## See ?effectsTheme for details.

## The Commander GUI is launched only in interactive sessions

##
## Attaching package: 'Rcmdr'

## The following object is masked from 'package:base':
##
##     errorCondition
```

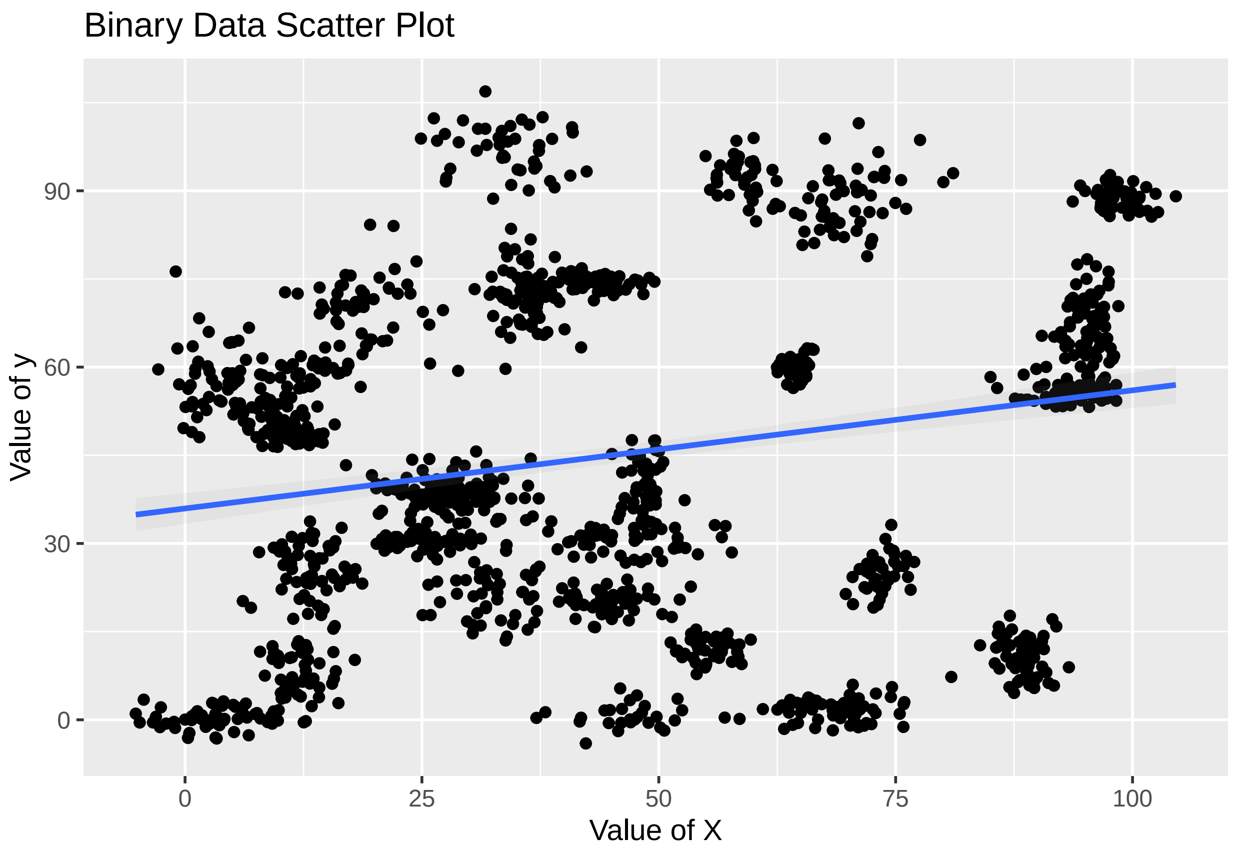
##Include all of your answers in a R Markdown report. ##Regression algorithms are used to predict numeric quantity while classification algorithms predict categorical outcomes. A spam filter is an example use case for a classification algorithm. The input dataset is emails labeled as either spam (i.e. junk emails) or ham (i.e. good emails). The classification algorithm uses features extracted from the emails to learn which emails fall into which category. ##In this problem, you will use the nearest neighbors algorithm to fit a model on two simplified datasets. The first dataset (found in binary-classifier-data.csv) contains three variables; label, x, and y. The label variable is either 0 or 1 and is the output we want to predict using the x and y variables (You worked with this dataset last week!). The second dataset (found in trinary-classifier-data.csv) is similar to the first dataset except that the label variable can be 0, 1, or 2.

```
##   label        x        y
## 1     0 70.88469 83.17702
## 2     0 74.97176 87.92922
## 3     0 73.78333 92.20325
## 4     0 66.40747 81.10617
## 5     0 69.07399 84.53739
## 6     0 72.23616 86.38403
```

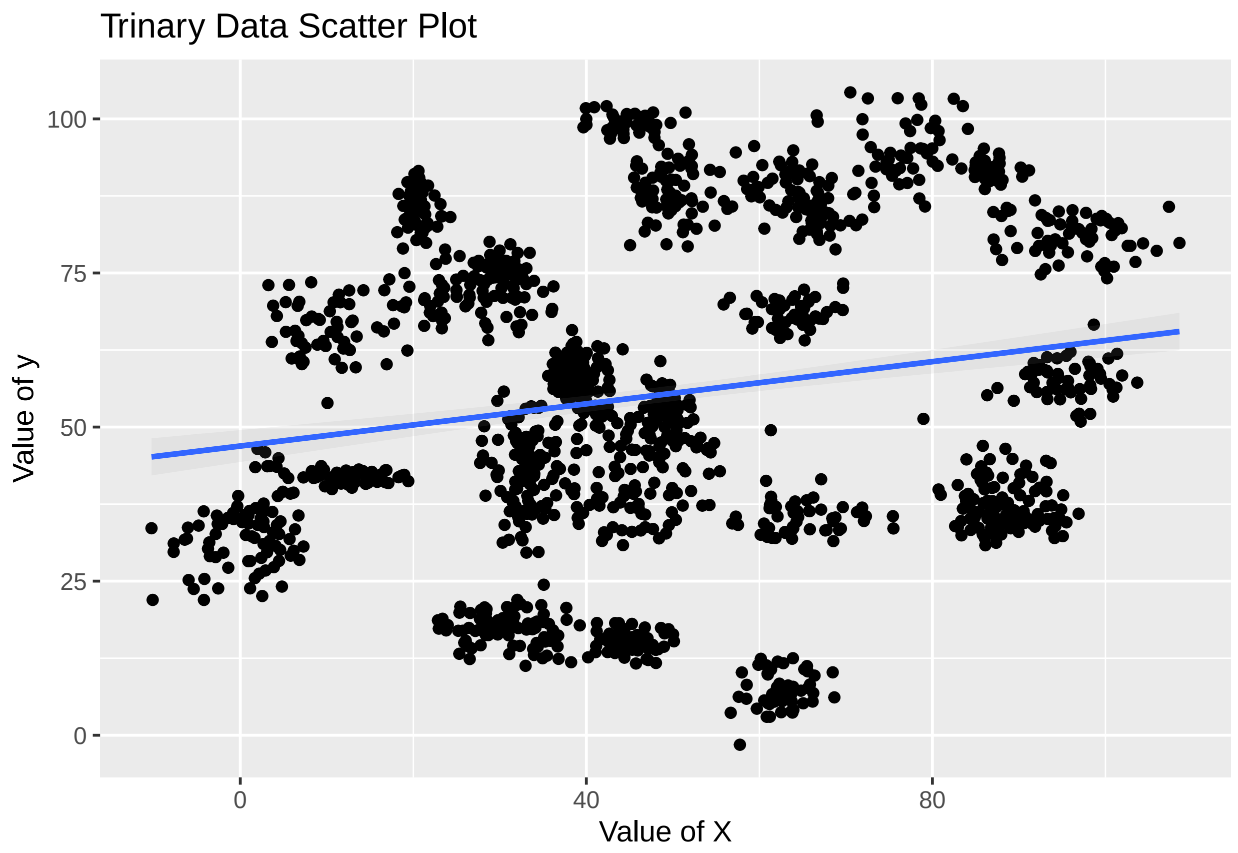
```
##   label        x        y
## 1     0 30.08387 39.63094
## 2     0 31.27613 51.77511
## 3     0 34.12138 49.27575
## 4     0 32.58222 41.23300
## 5     0 34.65069 45.47956
## 6     0 33.80513 44.24656
```

##Note that in real-world datasets, your labels are usually not numbers, but text-based descriptions of the categories (e.g. spam or ham). In practice, you will encode categorical variables into numeric values. #Plot the data from each dataset using a scatter plot.

```
## `geom_smooth()` using formula 'y ~ x'
```



```
## `geom_smooth()` using formula 'y ~ x'
```



###The k nearest neighbors algorithm categorizes an input value by looking at the labels for the k nearest points and assigning a category based on the most common label. In this problem, you will determine which points are nearest by calculating the Euclidean distance between two points. As a refresher, the Euclidean distance between two points: ##Fitting a model is when you use the input data to create a predictive model. There are various metrics you can use to determine how well your model fits the data. For this problem, you will focus on a single metric, accuracy. Accuracy is simply the percentage of how often the model predicts the correct result. If the model always predicts the correct result, it is 100% accurate. If the model always predicts the incorrect result, it is 0% accurate.

```
## [1] 1048
```

```
## [1] 450
```

```
## [1] 1048
```

```
## [1] 450
```

```
## [1] 1097
```

```
## [1] 471
```

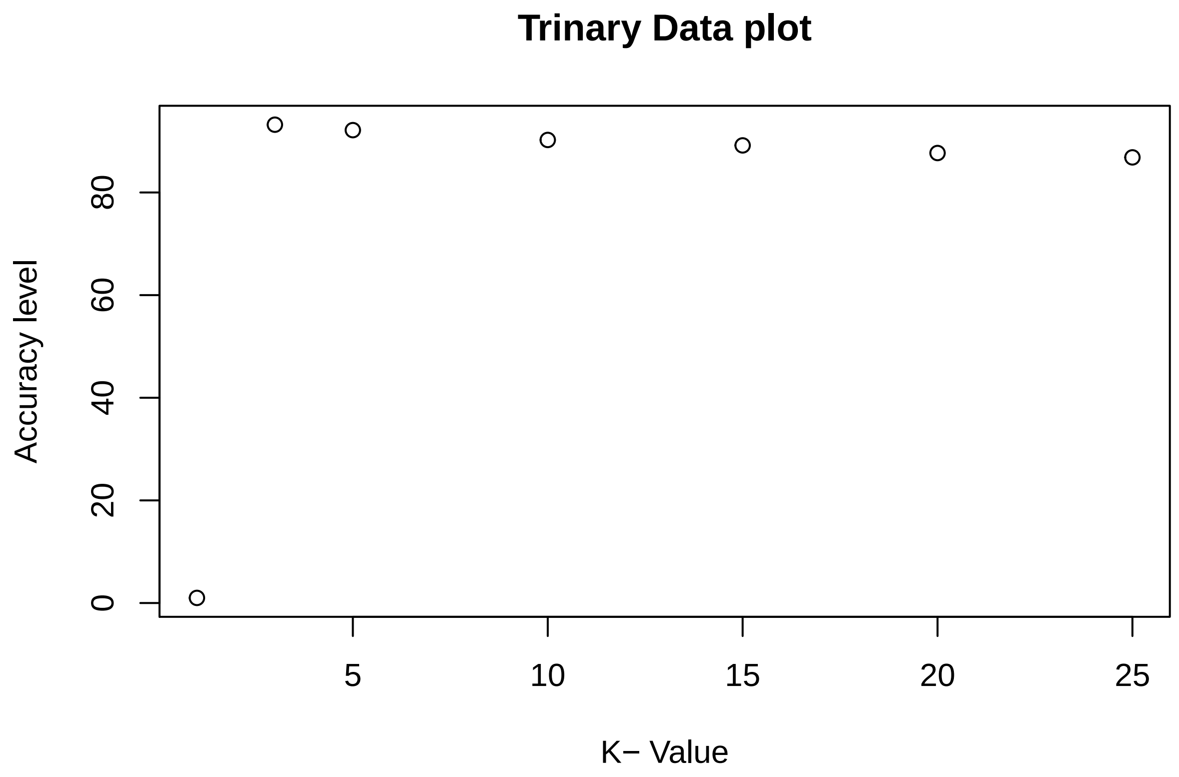
```
## [1] 1097
```

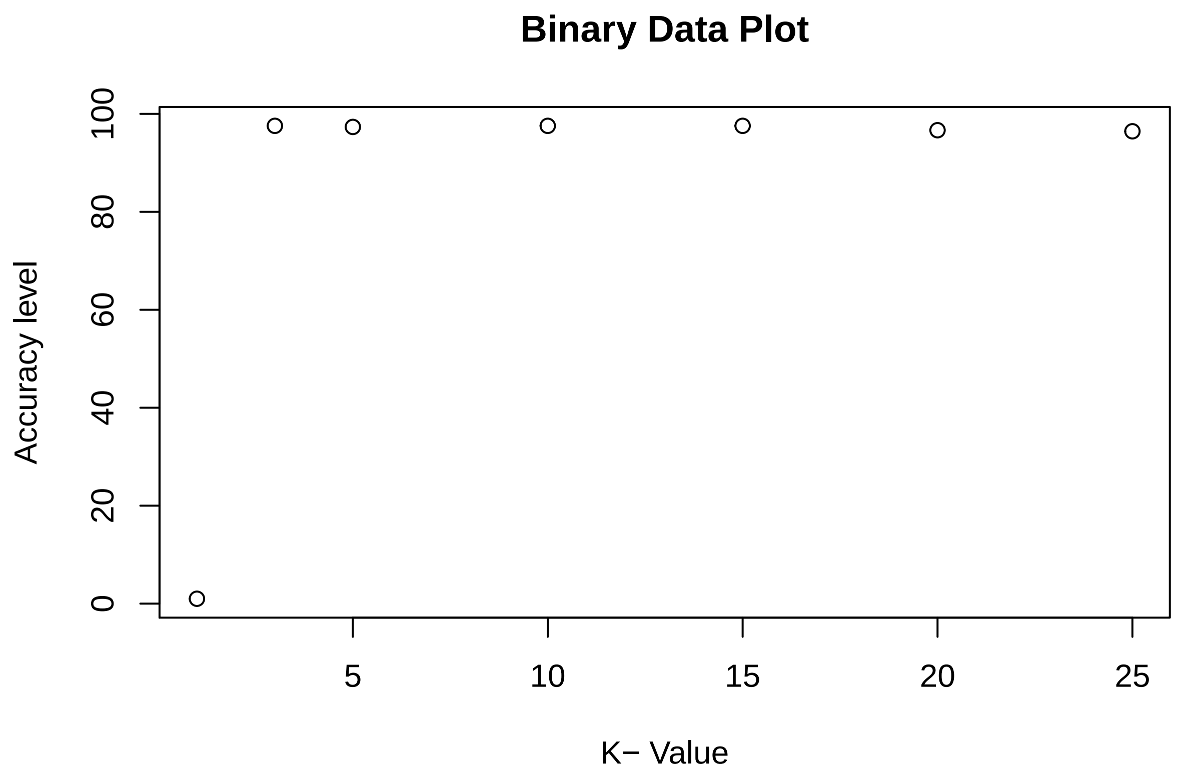
```
## [1] 471
```

##Fit a k nearest neighbors' model for each dataset for k=3, k=5, k=10, k=15, k=20, and k=25. Compute the accuracy of the resulting models for each value of k. Plot the results in a graph where the x-axis is the different values of k and the y-axis is the accuracy of the model.

```
## 3 = 97.55556 5 = 97.33333 10 = 97.55556 15 = 97.55556 20 = 96.66667 25 = 96.44444
```

```
## 3 = 93.20594 5 = 92.14437 10 = 90.23355 15 = 89.17197 20 = 87.68577 25 = 86.83652
```





###Looking back at the plots of the data, do you think a linear classifier would work well on these datasets? #How does the accuracy of your logistic regression classifier from last week compare? Why is the accuracy different between these two methods? last weeks logistic regression accuracy was 57%. Linear classifier Accuracy is greater than 80%, approaching high 90%s deoending on k size. Since the data is about predicting a label (classification) not a quantity(regression) in the current data set. Because we are looking to predict the label, linear classifer models accuracy is higher in comparision to logistic regression model from last week.

###Clustering ##These assignments are here to provide you with an introduction to the "Data Science" use for these tools. This is your future. It may seem confusing and weird right now but it hopefully seems far less so than earlier in the semester. Attempt these homework assignments. You will not be graded on your answer but on your approach. This should be a, "Where am I on learning this stuff" check. If you can't get it done, please explain why. ##Remember to submit this assignment in an R Markdown report. ##Labeled data is not always available. For these types of datasets, you can use unsupervised algorithms to extract structure. The k-means clustering algorithm and the k nearest neighbor algorithm both use the Euclidean distance between points to group data points. The difference is the k-means clustering algorithm does not use labeled data.

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.2     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x purrr::%@%()         masks rlang::%@%()
## x purrr::as_function() masks rlang::as_function()
## x dplyr::filter()      masks stats::filter()
## x purrr::flatten()     masks rlang::flatten()
## x purrr::flatten_chr() masks rlang::flatten_chr()
## x purrr::flatten_dbl() masks rlang::flatten_dbl()
## x purrr::flatten_int() masks rlang::flatten_int()
## x purrr::flatten_lgl() masks rlang::flatten_lgl()
## x purrr::flatten_raw() masks rlang::flatten_raw()
## x purrr::invoke()      masks rlang::invoke()
## x dplyr::lag()         masks stats::lag()
## x purrr::list_along()  masks rlang::list_along()
## x purrr::modify()      masks rlang::modify()
## x purrr::prepend()     masks rlang::prepend()
## x dplyr::recode()      masks car::recode()
## x purrr::some()        masks car::some()
## x purrr::splice()      masks rlang::splice()
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

##In this problem, you will use the k-means clustering algorithm to look for patterns in an unlabeled dataset. The dataset for this problem is found at data/clustering-data.csv.

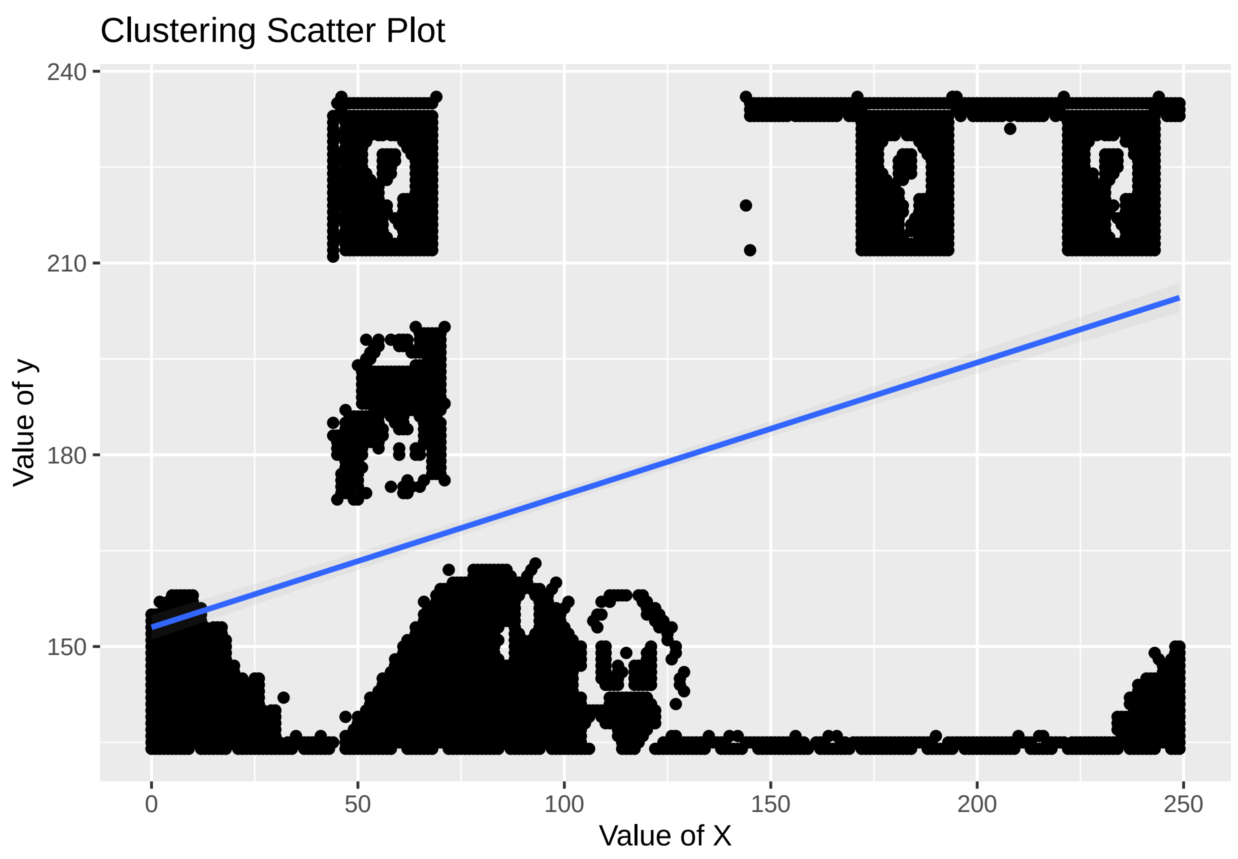
```
## 'data.frame':    4022 obs. of  2 variables:
##  $ x: int  46 69 144 171 194 195 221 244 45 47 ...
##  $ y: int  236 236 236 236 236 236 236 236 235 235 ...
```

```
## [1] "x" "y"
```

```
##            x        y
## 1 -0.8482235 1.561107
## 2 -0.5415045 1.561107
## 3  0.4586659 1.561107
## 4  0.8187273 1.561107
## 5  1.1254462 1.561107
## 6  1.1387818 1.561107
```

#Plot the dataset using a scatter plot.

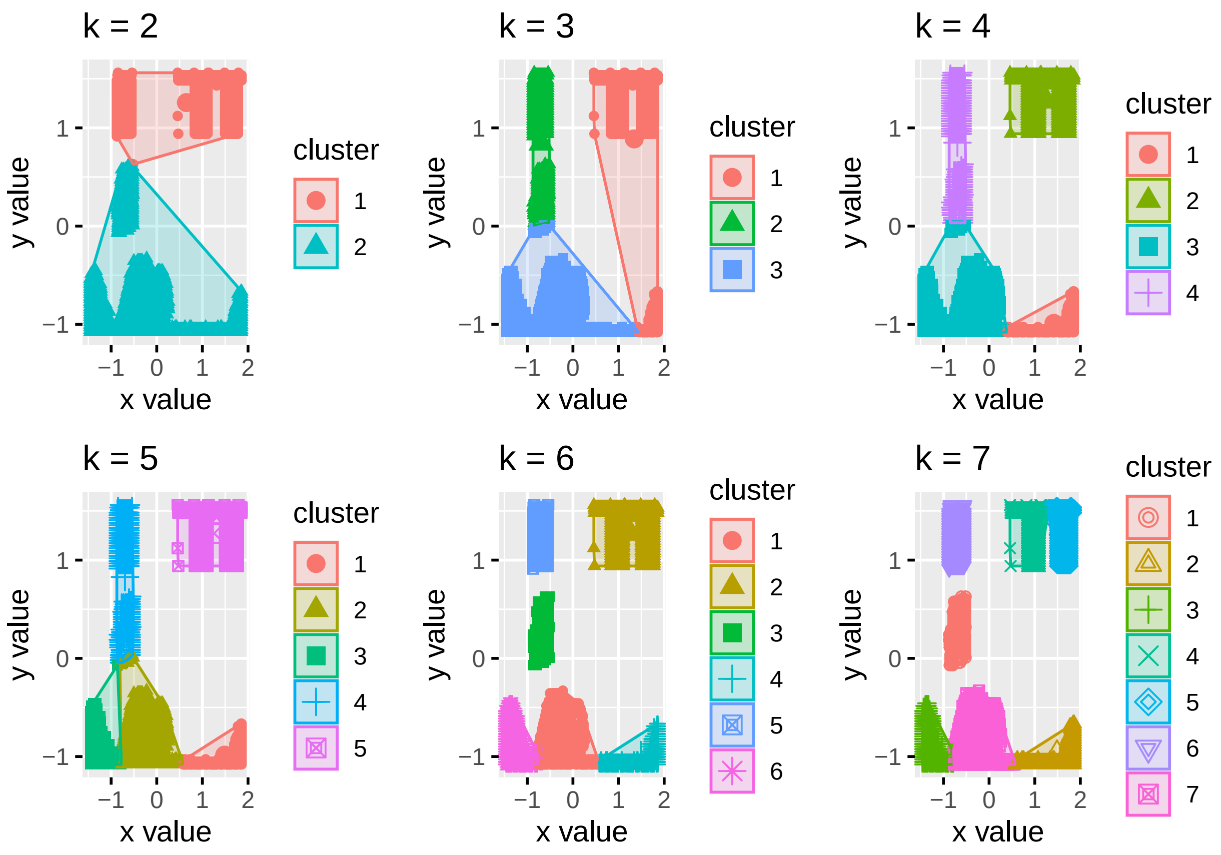
```
## `geom_smooth()` using formula 'y ~ x'
```



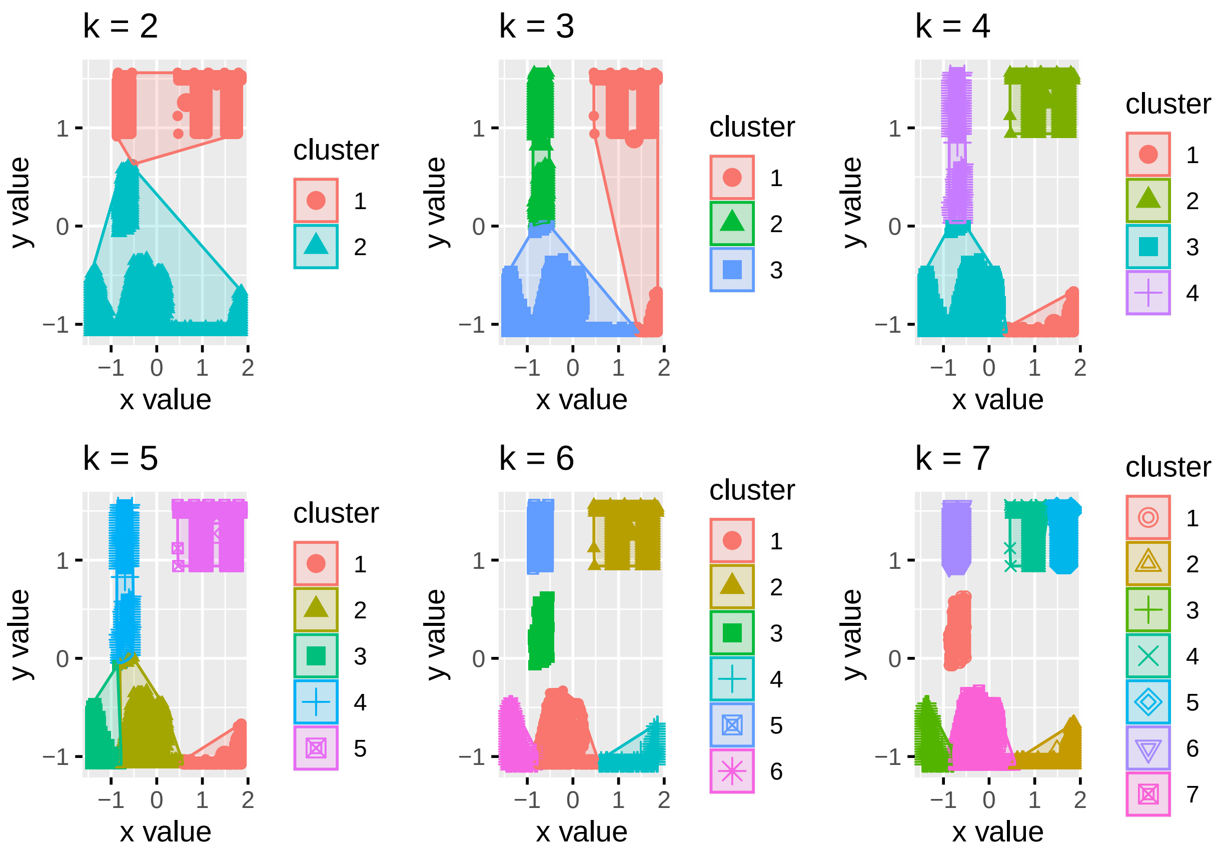
#Fit the dataset using the k-means algorithm from k=2 to k=12. Create a scatter plot of the resultant clusters for each value of k.

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

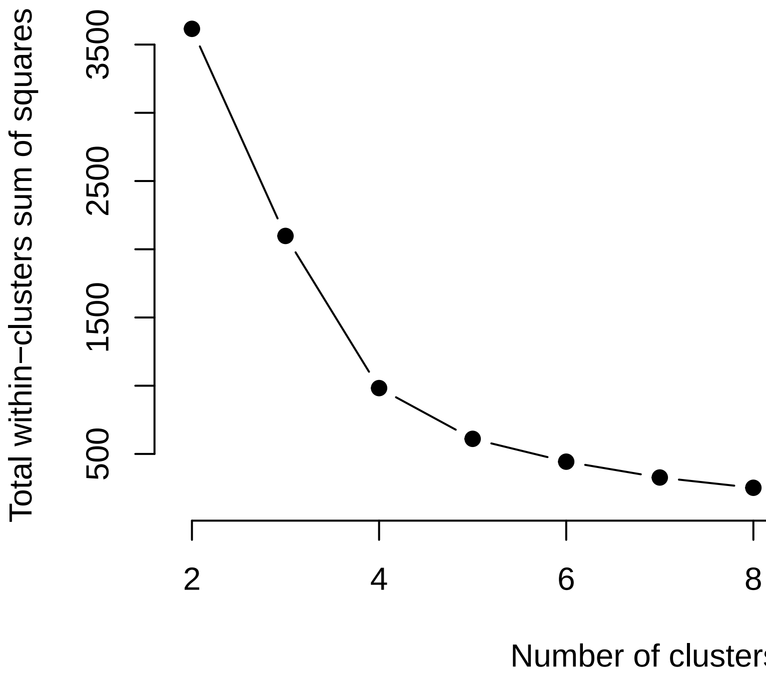


k = 2
k = 3
k = 4
k = 5
k = 6
k = 7
y value
x value
cluster
1
2
3
4
5
6
7

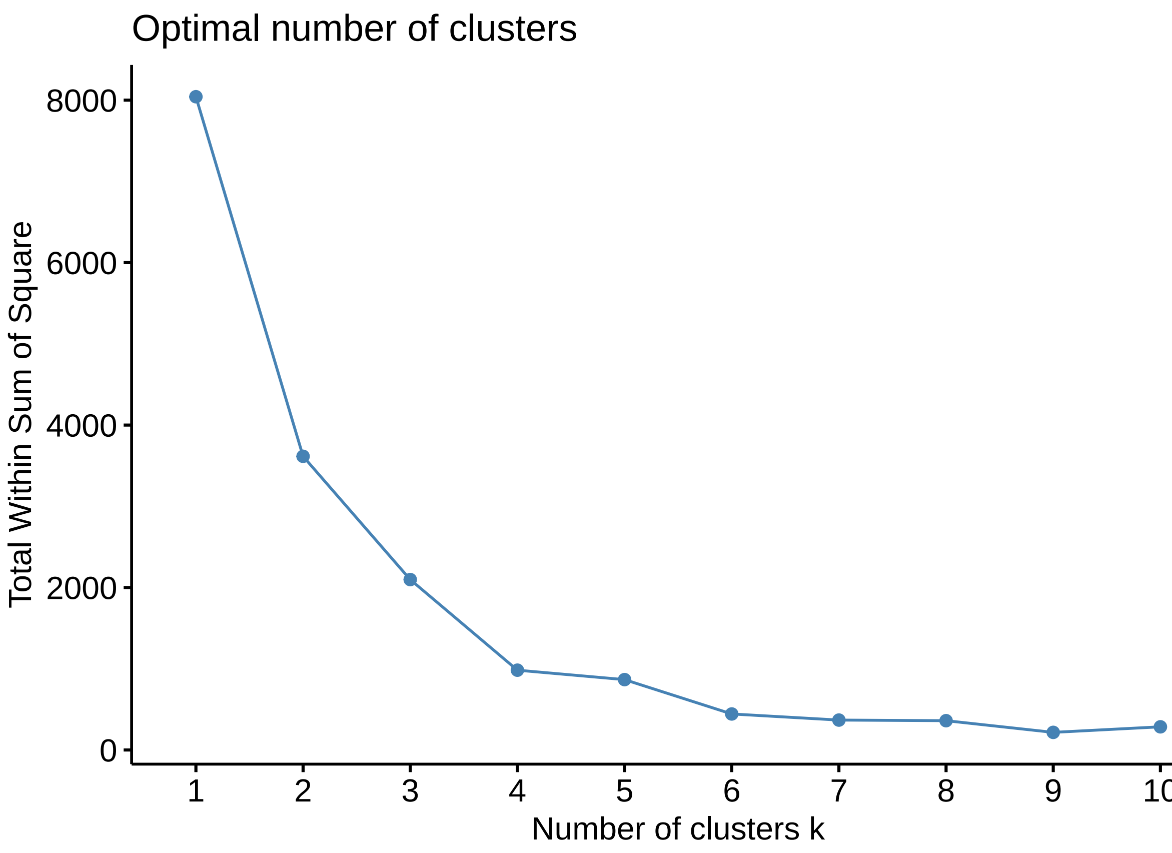


#As k-means is an unsupervised algorithm, you cannot compute the accuracy as there are no correct values to compare the output to. Instead, you will use the average distance from the center of each cluster as a measure of how well the model fits the data. To calculate this metric, simply compute the distance of each data point to the center of the cluster it is assigned to and take the average value of all of those distances.

##Calculate this average distance from the center of each cluster for each value of k and plot it as a line chart



where k is the x-axis and the average distance is the y-axis.
##One way of determining the "right" number of clusters is to look at the graph of k versus average distance and finding the "elbow point". Looking at the graph you generated in the previous example, what is the elbow



point for this dataset?

elbow point for the data set is 4