Question 3.1

Using the same data set (`credit_card_data.txt` or `credit_card_data-headers.txt`) as in Question 2.2, use the `ksvm` or `kknn` function to find a good classifier:

(a) using cross-validation (do this for the k-nearest-neighbors model; SVM is optional); and
(b) splitting the data into training, validation, and test data sets (pick either KNN or SVM; the other is optional).

To answer this question, I utilized the packages caret, kknn, kernlab, ggplot2, and shiny to demonstrate model selection and cross-validation. My program first splits the data into training, validation, and test sets (or uses caret's 10-fold cross-validation) and then systematically varies hyperparameters for kNN (the number of neighbors k) and SVM (the penalty parameter C) to identify which options yield the highest accuracy. The ggplot2 plots help visualize how accuracy changes over different k and C values, while shiny serves as a primitive frontend for the data.

Here is the final output to my console:

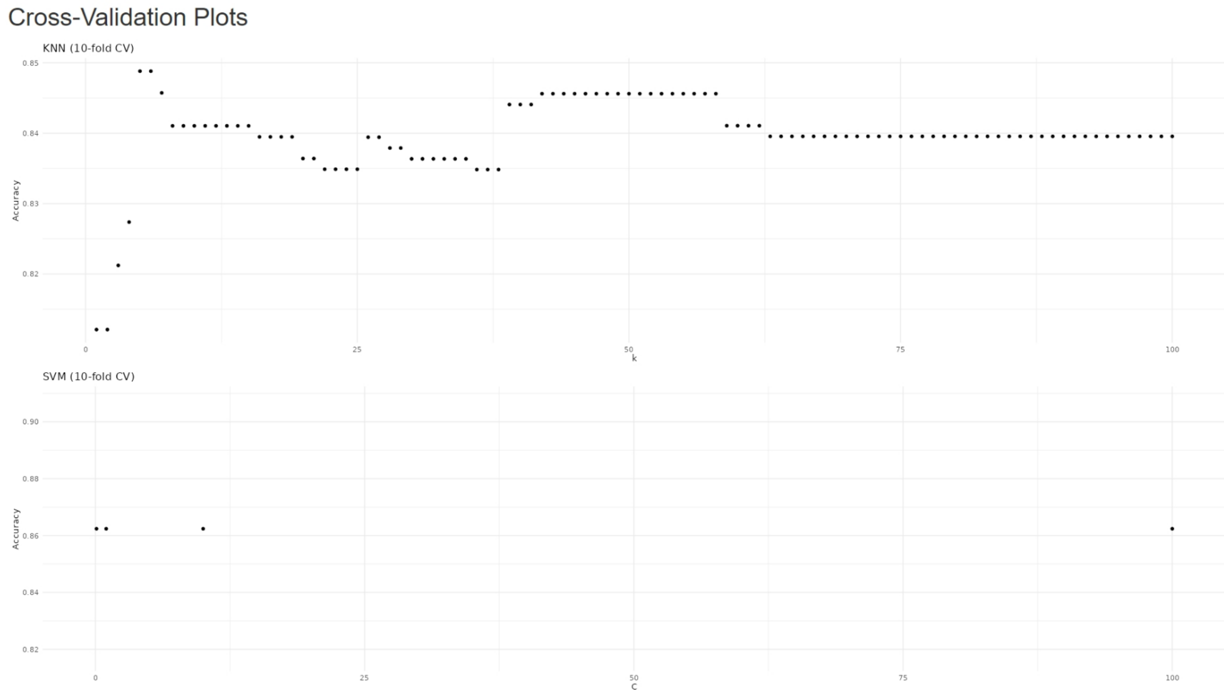
```
(base) user@DESKTOP-FMSNMOE:~/Repos/R/AnalyticsModeling/HW2$ Rscript 3dot1Classifier.R
Loading required package: ggplot2
Loading required package: lattice

Attaching package: 'kknn'

The following object is masked from 'package:caret':

    contr.dummy


Attaching package: 'kernlab'

The following object is masked from 'package:ggplot2':

    alpha


--- KNN CROSS-VALIDATION RESULTS ---
k-Nearest Neighbors

654 samples
 10 predictor
  2 classes: 'Class0', 'Class1'

Pre-processing: centered (10), scaled (10)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 588, 589, 589, 588, 588, 590, ...
Resampling results across tuning parameters:
```

kmax	Accuracy	Kappa
1	0.8121154	0.6184220
2	0.8121154	0.6184220
3	0.8212296	0.6374644
4	0.8273834	0.6505935
5	0.8488061	0.6957836
6	0.8488061	0.6957836
7	0.8457292	0.6898036
8	0.8410417	0.6807152
9	0.8410417	0.6807152
10	0.8410417	0.6807152
11	0.8410417	0.6807152
12	0.8410417	0.6809005
13	0.8410417	0.6809005
14	0.8410417	0.6809005
15	0.8410417	0.6809005
16	0.8394792	0.6776647
17	0.8394792	0.6776647
18	0.8394792	0.6776647
19	0.8394792	0.6776647
20	0.8364022	0.6712741
21	0.8364022	0.6712741
22	0.8348871	0.6679912
23	0.8348871	0.6679912
24	0.8348871	0.6679912
25	0.8348871	0.6679912
26	0.8394325	0.6769782
27	0.8394325	0.6769782
28	0.8378941	0.6732430
29	0.8378941	0.6732430
30	0.8363556	0.6700155
31	0.8363556	0.6700155
32	0.8363556	0.6700155
33	0.8363556	0.6700155
34	0.8363556	0.6700155
35	0.8363556	0.6700155
36	0.8348405	0.6668651
37	0.8348405	0.6668651
38	0.8348405	0.6668651
39	0.8440712	0.6850472
40	0.8440712	0.6850472
41	0.8440712	0.6850472
42	0.8456097	0.6878124
43	0.8456097	0.6878124
44	0.8456097	0.6878124
45	0.8456097	0.6878124
46	0.8456097	0.6878124

47	0.8456097	0.6878124
48	0.8456097	0.6878124
49	0.8456097	0.6878124
50	0.8456097	0.6878124
51	0.8456097	0.6878124
52	0.8456097	0.6878124
53	0.8456097	0.6878124
54	0.8456097	0.6878124
55	0.8456097	0.6878124
56	0.8456097	0.6878124
57	0.8456097	0.6878124
58	0.8456097	0.6878124
59	0.8410642	0.6781723
60	0.8410642	0.6781723
61	0.8410642	0.6781723
62	0.8410642	0.6781723
63	0.8395491	0.6749687
64	0.8395491	0.6749687
65	0.8395491	0.6749687
66	0.8395491	0.6749687
67	0.8395491	0.6749687
68	0.8395491	0.6749687
69	0.8395491	0.6749687
70	0.8395491	0.6749687
71	0.8395491	0.6749687
72	0.8395491	0.6749687
73	0.8395491	0.6749687
74	0.8395491	0.6749687
75	0.8395491	0.6749687
76	0.8395491	0.6749687
77	0.8395491	0.6749687
78	0.8395491	0.6749687
79	0.8395491	0.6749687
80	0.8395491	0.6749687
81	0.8395491	0.6749687
82	0.8395491	0.6749687
83	0.8395491	0.6749687
84	0.8395491	0.6749687
85	0.8395491	0.6749687
86	0.8395491	0.6749687
87	0.8395491	0.6749687
88	0.8395491	0.6749687
89	0.8395491	0.6749687
90	0.8395491	0.6749687
91	0.8395491	0.6749687
92	0.8395491	0.6749687
93	0.8395491	0.6749687

```
  94   0.8395491  0.6749687
  95   0.8395491  0.6749687
  96   0.8395491  0.6749687
  97   0.8395491  0.6749687
  98   0.8395491  0.6749687
  99   0.8395491  0.6749687
 100   0.8395491  0.6749687

Tuning parameter 'distance' was held constant at a value of 2
Tuning
 parameter 'kernel' was held constant at a value of rectangular
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were kmax = 6, distance = 2 and kernel
 = rectangular.

--- SVM CROSS-VALIDATION RESULTS ---
Support Vector Machines with Linear Kernel

654 samples
 10 predictor
  2 classes: 'Class0', 'Class1'

Pre-processing: centered (10), scaled (10)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 588, 589, 588, 589, 589, 589, ...
Resampling results across tuning parameters:

  C      Accuracy   Kappa
    0.1  0.8624009  0.7270535
    1.0  0.8624009  0.7270535
   10.0  0.8624009  0.7270535
  100.0  0.8624009  0.7270535

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 0.1.

Listening on http://127.0.0.1:6855
```

My program ‘3dot1Classifier.R’ effectively utilizes caret’s cross validation methods to train models based on my pre-partitioned data and to evaluate model parameters to maximize accuracy. Ideal conditions seem to be kmax = 6 (for kNN) and C = 0.1 (for SVM).



Question 4.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

For a few months during my undergraduate years, I worked as a Student Ambassador for our campus' dining center. While working there, occasionally we were tasked with surveying different people within the dining hall as they exited. This was an extremely unoptimized process that required us ambassadors to ask any customer who left the dining center if they could sit down with us to take a survey. At the same time, the dining center had sensors in place to track how many people entered and left, yet we never leveraged that data to refine our approach or seek more targeted responses. A clustering model would have been appropriate here because it can group diners based on common patterns or behaviors without needing predefined categories. This could reveal natural segments of people (for instance, those who dine mainly at breakfast or late night, individuals who use meal plans exclusively, or those who only drop by on weekends), and it could guide ambassadors to survey specific groups more effectively.

In constructing such a model, we might use factors like visitors' entry and exit times, payment preferences, how frequently they patronize the dining hall, which days of the week are most popular for them, size of group upon entry, and whether they live on or off campus.

By applying these variables, the dining center could optimize staffing, prepare more accurate meal quantities, and direct surveys toward underrepresented clusters. This approach would likely result in higher-quality feedback, less wasted time, and a more engaging dining experience for everyone involved. It may also help us to identify which clusters are most partial to being surveyed.

Question 4.2

The *iris* data set `iris.txt` contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris). *The response values are only given to see how well a specific method performed and should not be used to build the model.*

Use the R function `kmeans` to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

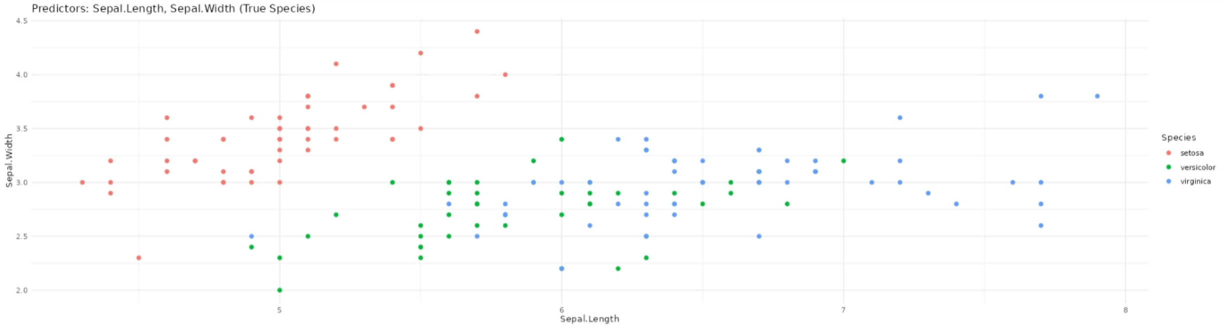
(base) user@DESKTOP-FMSNMOE:~/Repos/R/AnalyticsModeling/HW2$ Rscript 4dot2Iris.R
Subset: Sepal.Length, Sepal.Width | Best k = 30 | Best-k Accuracy ~ 0.84 | k=3 Accuracy ~ 0.82
Subset: Petal.Length, Petal.Width | Best k = 15 | Best-k Accuracy ~ 0.9733333 | k=3 Accuracy ~ 0.96
Subset: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width | Best k = 31 | Best-k Accuracy ~ 0.9866667 | k=3 Accuracy ~ 0.8933333
Warning messages:
1: did not converge in 200 iterations
2: did not converge in 200 iterations
3: did not converge in 200 iterations

My code scans k=1:40 to find the best alignment with actual species, then reports that k value and accuracy. It also checks k=3 separately to compare its accuracy to the algorithmic choice. The Shiny app shows three scatter plots: one colored by true species, one by the clusters at the "best" k, and one by k=3. K=3 was chosen due to there being 3 actual different species in the data set. This process was iteratively applied to all variations of variable columns producible by the original dataset.

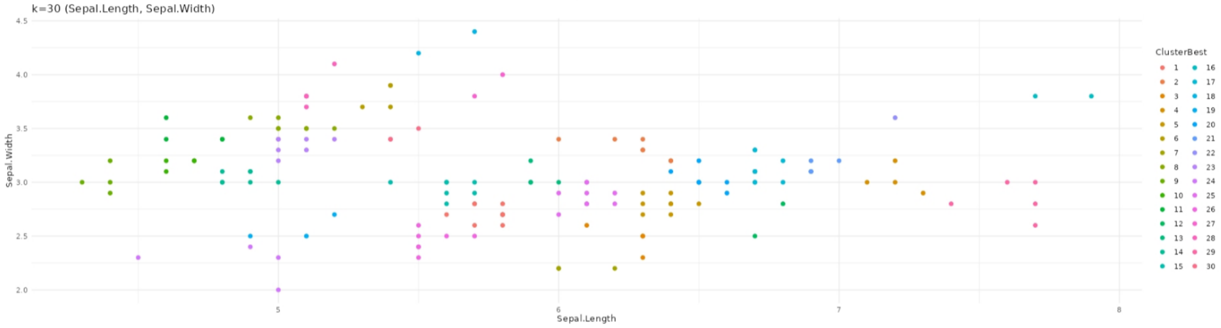
Models performance was plotted using ggplot2 and shiny.

Sepal.Length, Sepal.Width | Petal.Length, Petal.Width | Sepal.Length, Sepal.Width, Petal.Length, Petal.Width

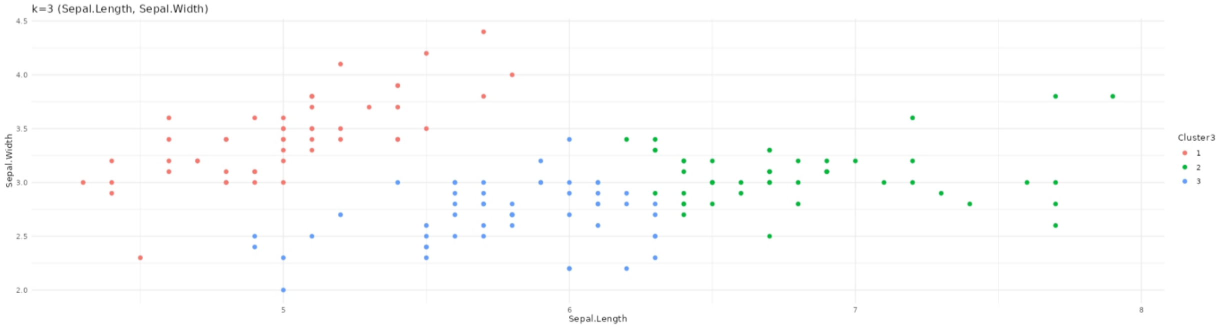
Predictors: Sepal.Length, Sepal.Width (True Species)



k=30 (Sepal.Length, Sepal.Width)

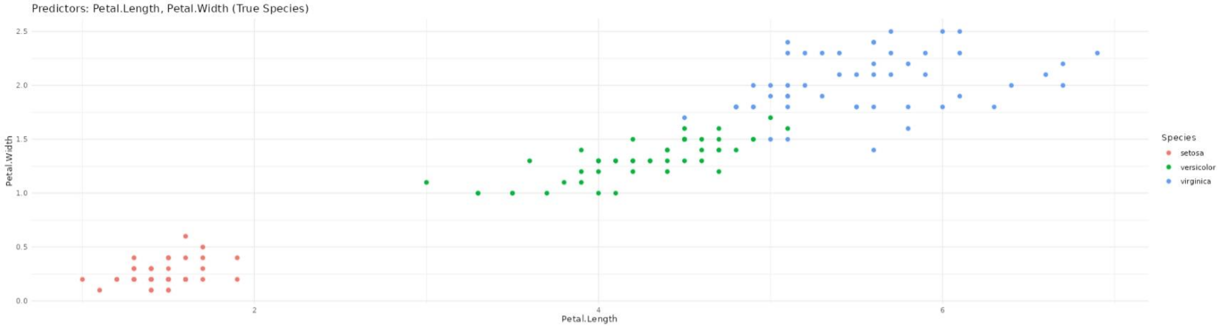


k=3 (Sepal.Length, Sepal.Width)

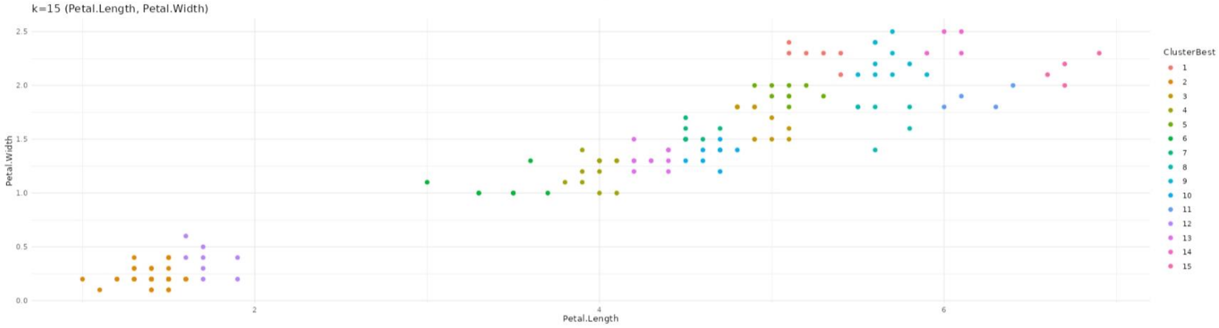


Sepal.Length, Sepal.Width | Petal.Length, Petal.Width | Sepal.Length, Sepal.Width, Petal.Length, Petal.Width

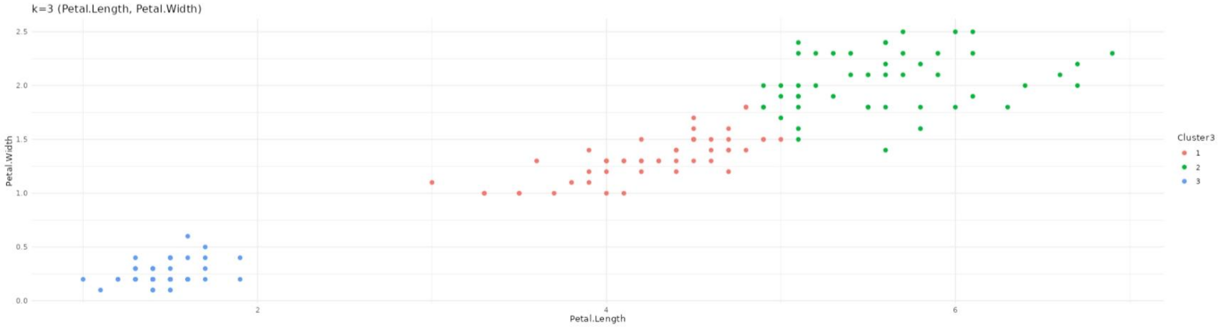
Predictors: Petal.Length, Petal.Width (True Species)



k=15 (Petal.Length, Petal.Width)



k=3 (Petal.Length, Petal.Width)



Although the 'greatest' accuracy settings were at higher k values such as k=15/31/30, I would argue that these clustering models result in overfitting that makes the reported clusters difficult to understand. Depending on the original variables used to form clusters, k=3 had varying success. The clustering model with the greatest fit to the original source data seems to be k=3 for the subset: Petal.Length, Petal.Width. Performing at 96% accuracy, the output graph very closely matches the original dataset.

References:

ChatGPT's o1 Model
Stack Overflow
Shiny documentation
Caret documentation