DSC 441 – FUNDAMENTALS OF DATA SCIENCE

HOMEWORK – 5

NAME – Goutham Selvakumar

a. Data gathering and integration

The first part is to get the data you will use. You may use anything that has not been used in an assignment or tutorial. It must have at least 100 data points and must include both numerical and categorial (or ordinal) variables. I recommend keeping this relatively straightforward because data cleaning can take a lot of time if you choose a large, messy dataset. Kaggle (https://www.kaggle.com/datasets) and the University of California at Irvine (UCI) (https://archive.ics.uci.edu/ml/index.php) maintain collections of datasets, some even telling you if they are good examples for testing specific machine learning techniques. You may also choose to join together more than one dataset, for example to merge data on health outcomes by US state with a dataset on food statistics per state. Merging data is not required and will earn you a bonus point in this step.

```r
#Import insurance dataset
library(readxl)
insurance <- read_csv("insurance.csv")
head(insurance)
```

A tibble: 6 x 7

age <dbl>	sex <chr>	bmi <dbl>	children <dbl>	smoker <chr>	region <chr>	charges <dbl>
19	female	27.900	0	yes	southwest	16884.924
18	male	33.770	1	no	southeast	1725.552
28	male	33.000	3	no	southeast	4449.462
33	male	22.705	0	no	northwest	21984.471
32	male	28.880	0	no	northwest	3866.855
31	female	25.740	0	no	southeast	3756.622

6 rows

I have decided to use the insurance dataset to predict the smokers based on the data provided I'd like work off of the hypothesis that charges will likely increase if you're a smoker. This will set the tone for the rest of my analysis.

b. Data Exploration

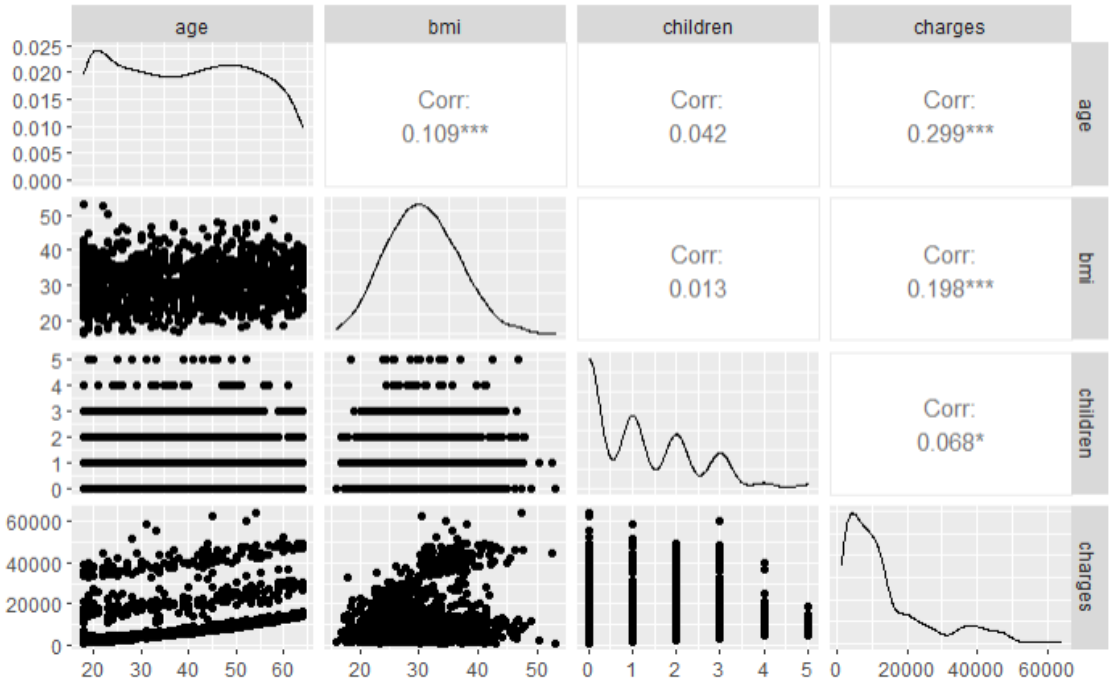
Using data exploration to understand what is happening is important throughout the pipeline, and is not limited to this step. However, it is important to use some exploration early on to make sure you understand your data. You must at least consider the distributions of each variable and at least some of the relationships between pairs of variables.

```
summary(insurance)
```

```
      age            sex                 bmi           children       smoker             region             charges     
 Min.   :18.00   Length:1338        Min.   :15.96   Min.   :0.000   Length:1338        Length:1338        Min.   : 1122  
 1st Qu.:27.00   Class :character   1st Qu.:26.30   1st Qu.:0.000   Class :character   Class :character   1st Qu.: 4740  
 Median :39.00   Mode  :character   Median :30.40   Median :1.000   Mode  :character   Mode  :character   Median : 9382  
 Mean   :39.21                      Mean   :30.66   Mean   :1.095                                         Mean   :13270  
 3rd Qu.:51.00                      3rd Qu.:34.69   3rd Qu.:2.000                                         3rd Qu.:16640  
 Max.   :64.00                      Max.   :53.13   Max.   :5.000                                         Max.   :63770  
```

There are no NA's in this dataset. Sex, region, and smokers are categorical, while the rest of the dataset is numeric. Now, let's perform some more exploratory tasks and find existing relationships. First, I will create a scatterplot matrix of the numerical variables.

```
insurance %>% select(age, bmi, children, charges) %>% ggpairs()
```

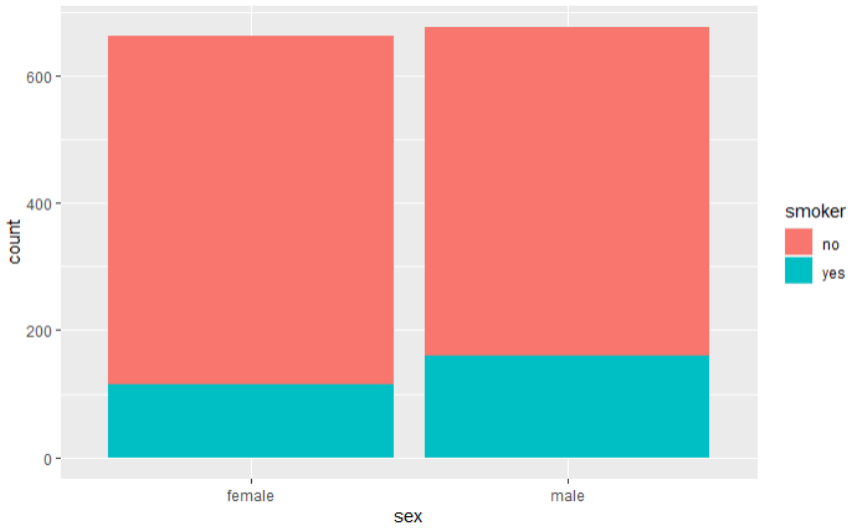


Now, I will list a few things worth pointing out here:

1. While there is outliers, we can see graphically that charges generally increase with age. The Pearson-correlation coefficient also shows a positive correlation between charges and age.
2. The trend between BMI and charges isn't that clear graphically, but we can see some that there is definitely an increase in charges for some individuals after hitting the age of 30. The Pearson-correlation also proves a somewhat positive correlation. This will be interesting to discover later based on sex and smoker-status.
3. Note the somewhat positive correlation number between age and BMI, although this is unclear graphically.

Now, it will be interesting to view the number of female to men whom are smokers to nonsmokers:

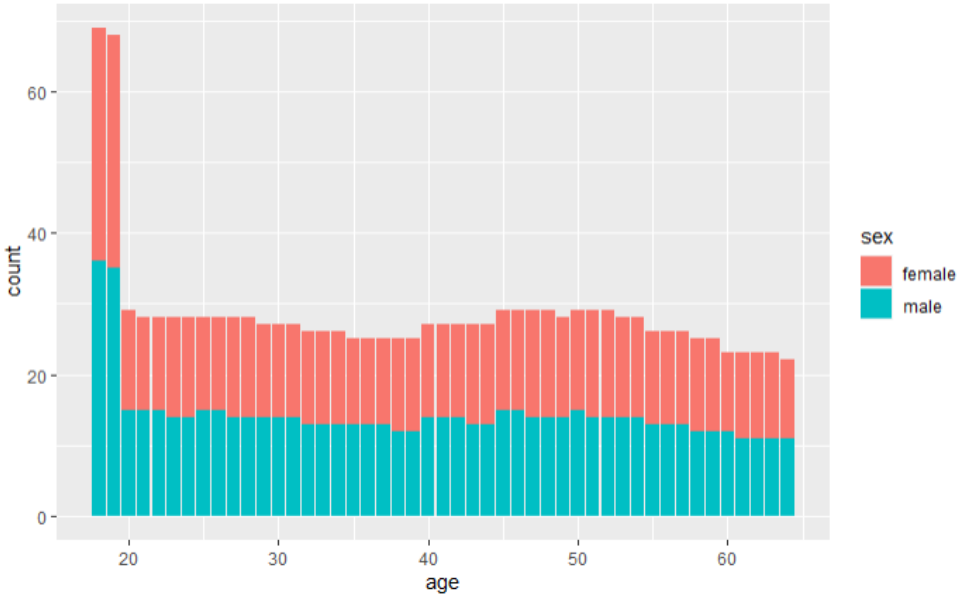
```
#Convert to dataframe
df <- as.data.frame(insurance)
#Create ggplot object
p <- ggplot(insurance, aes(x=sex, fill=smoker))
p + geom_bar(position = "stack")
```



We can see here that, as expected, there is slightly more smokers among men than there is smokers among women. Another important thing to note is that there is almost equally as many men represented in the dataset as there is women.

Let's see the actual ages represented in this dataset, as well as sex per age, and try to see if it's balanced:

```
p <- ggplot(df, aes(x=age, fill=sex))
p + geom_bar(position = "stack")
```



We can see here that there is such a high representation of ages under 20. To eliminate bias towards such younger age groups, it may help to remove all input from ages under 20.

Let's quickly view the number of smokers here:

```
insurance %>% group_by(smoker) %>% summarise("count"=n())
```

A tibble: 2 x 2

smoker <chr>	count <int>
no	1064
yes	274

2 rows

We have 274 smokers in this dataset.

Now, let's look at this graphically:

```
ggplot(insurance, aes(x=age, y=charges, color=smoker)) +
  geom_point()
```

Many of those smokers are the supposed 'outliers' here, in which they are charged much more for insurance than non-smokers. This is important to note, as non-smoker charges seem to be far more consistent and most likely predictable compared to non-smokers.

c. Data Cleaning

Don't forget – this can take a lot of the time of the whole process. Your cleaning process must ensure that there are no missing values and all outliers must be considered. It may be reasonable to just remove rows with missing values, however, if your data or small or that would change the distributions of the variables, that will not be adequate and you will need to consider other options, as discussed in the modules on cleaning.
Depending on your data and what you plan to do with it, you may also need to apply other processes we discussed. For example, clean up strings for consistency, deal with date formatting, change variable types between categorical and numeric, bin, smooth, group, aggregate or reshape. Make the case with visualization or by showing resulting summary statistics that your data are clean enough to continue with your analysis.

Now, let's do some data cleaning. Fortunately, this dataset seems to be mostly straightforward with no NA's (or) missing data as we saw in the previous step.

```
summary(insurance)
```

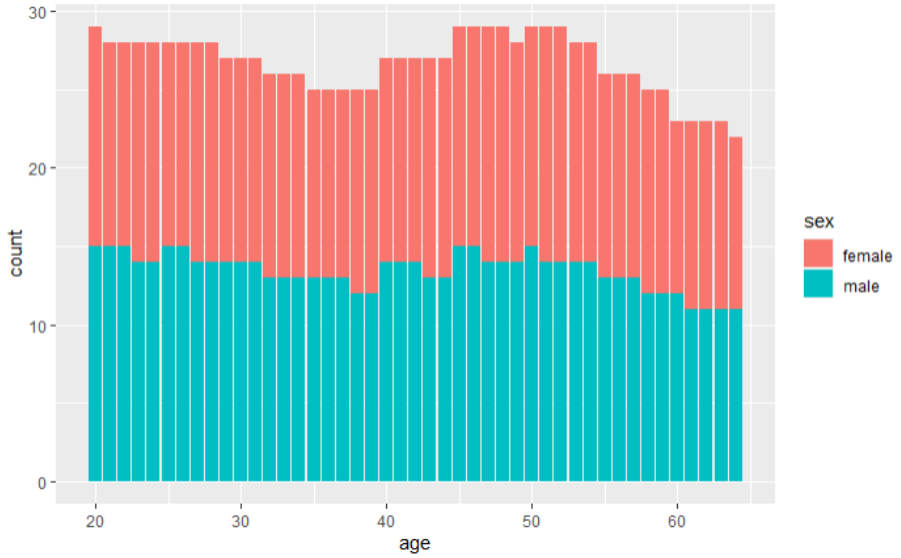
```
      age            sex                 bmi           children       smoker            region
 Min.   :18.00   Length:1338        Min.   :15.96   Min.   :0.000   Length:1338        Length:1338
 1st Qu.:27.00   Class :character   1st Qu.:26.30   1st Qu.:0.000   Class :character   Class :character
 Median :39.00   Mode  :character   Median :30.40   Median :1.000   Mode  :character   Mode  :character
 Mean   :39.21                      Mean   :30.66   Mean   :1.095
 3rd Qu.:51.00                      3rd Qu.:34.69   3rd Qu.:2.000
 Max.   :64.00                      Max.   :53.13   Max.   :5.000
    charges
 Min.   : 1122
 1st Qu.: 4740
 Median : 9382
 Mean   :13270
 3rd Qu.:16640
 Max.   :63770
```

Although, we did notice that there may be a bias towards ages younger than 20, so I will not remove all data points of those younger than 20.

```
#Exclude rows using subset function with condition of including ages greater than or equal to 20
insurance <- subset(insurance, age >=20)
```

Now, let's see how this looks graphically:

```
p <- ggplot(insurance, aes(x=age, fill=sex))
p + geom_bar(position = "stack")
```



It now looks much more balanced. The rest of the dataset seems to be good at this time and no additional cleaning is necessary.

d. Data Preprocessing

In some cases, preprocessing is absolutely necessary. It is rarely a bad idea. Make the case for what is and is not necessary given what you plan to do with the data. This could include making dummy variables, applying normalization, binning and/or smoothing, and other transformations (see course module).

I think we will most likely use the dummy variables, so I will go ahead and create those now. But first, I will remove the region as it may not be necessary.

```
#Remove region's column
df <- insurance
predictors <- df %>% select(-c(region))
head(predictors)
```

A tibble: 6 x 6

age <dbl>	sex <chr>	bmi <dbl>	children <dbl>	smoker <chr>	charges <dbl>
28	male	33.000	3	no	4449.462
33	male	22.705	0	no	21984.471
32	male	28.880	0	no	3866.855
31	female	25.740	0	no	3756.622
46	female	33.440	1	no	8240.590
37	female	27.740	3	no	7281.506

6 rows

Now the dummies:

```
#Create dummies
dummy <- dummyVars(smoker ~ ., data = predictors)
dummies <- as.data.frame(predict(dummy, newdata = predictors))
head(dummies)
```

Description: df [6 x 6]

	age <dbl>	sexfemale <dbl>	sexmale <dbl>	bmi <dbl>	children <dbl>	charges <dbl>
1	28	0	1	33.000	3	4449.462
2	33	0	1	22.705	0	21984.471
3	32	0	1	28.880	0	3866.855
4	31	1	0	25.740	0	3756.622
5	46	1	0	33.440	1	8240.590
6	37	1	0	27.740	3	7281.506

6 rows

e. Clustering

Remove any labels from your data and use clustering to discover any built-in structure. Use an appropriate method to determine the number of clusters. If your data have labels, compare the clusters to those labels. If not, visualize the clustering results by making a PCA projection and coloring the points by cluster assignment. Note that PCA only works for numerical variables, so if your data have just a few categoricals, you may skip them. If there are many, use dummy variables or choose a different method for making a projection. One way is to make the distance matrix first (we covered a method for distance matrices using categorical variables in the clustering tutorial) and then apply PCA to that matrix. This is actually a way to calculate an MDS projection, a very popular method.

I will rename the dummies as predictors:

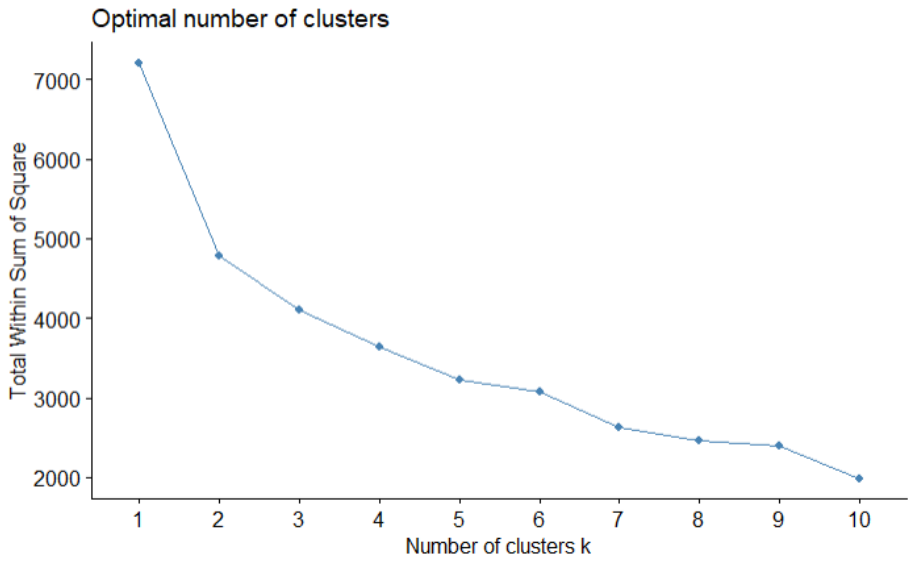
```
predictors <- dummies
```

Now, let's use the K-means to cluster the data:

```
#Center scale allows us to standardize the data
preproc <- preProcess(predictors, method =c("center", "scale"))
#We have to call predict to fit our data based on preprocessing
predictors <- predict(preproc, predictors)
```

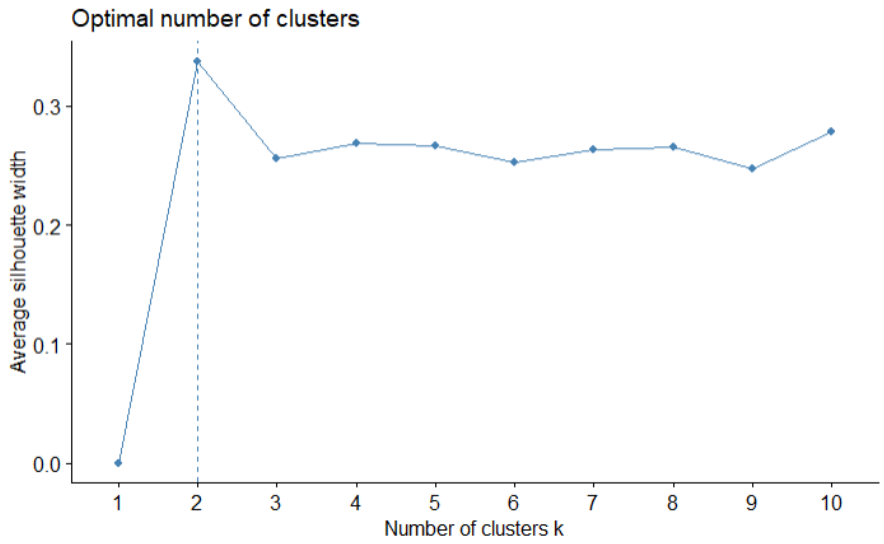
Now, find the knee:

```
#Find the knee
fviz_nbclust(predictors, kmeans, method = "wss")
```



Next, we can use the silhouette to find the different K values:

```
fviz_nbclust(predictors, kmeans, method = "silhouette")
```



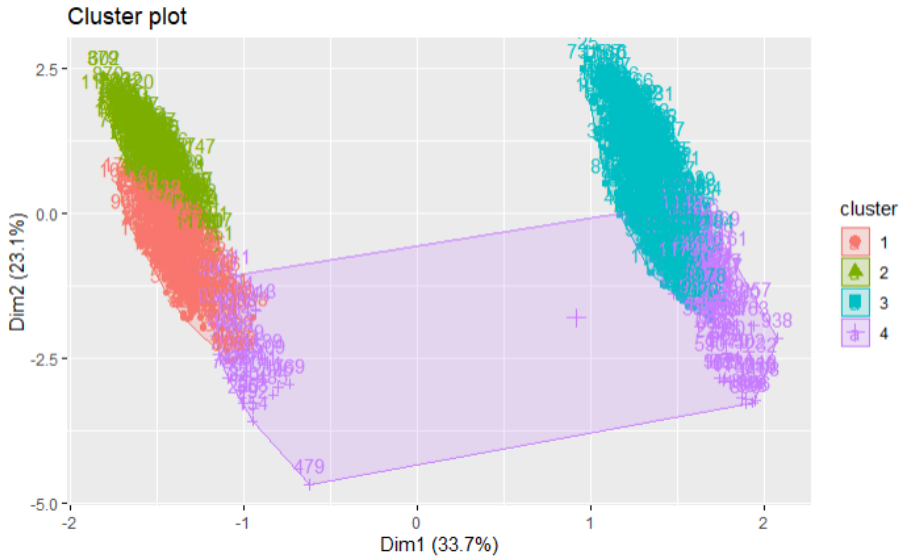
The knee suggests a K of 4 but the silhouette score suggests K=2. In this case, we will choose K of 4.

```
#Fit the data
fit <- kmeans(predictors, centers = 4, nstart = 25)
#Display the kmeans object information
fit
```

```
K-means clustering with 4 clusters of sizes 280, 276, 510, 135

Cluster means:
        age  sexfemale    sexmale         bmi     children      charges
1  0.8566914  1.0071025 -1.0071025  0.06847007 -0.072196373 -0.003991792
2 -0.8611210  1.0071025 -1.0071025 -0.35858721  0.027194427 -0.495706273
3 -0.0308715 -0.9921208  0.9921208 -0.04708594  0.007056144 -0.341240389
4  0.1002982 -0.3997584  0.3997584  0.76897985  0.067486588  2.316986898

Clustering vector:
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24
  3   3   3   2   1   2   3   1   3   1   3   1   4   1   3   3   4   1   2   4   3   1   1   1
 25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48
  3   4   4   3   4   1   3   4   4   2   2   3   2   3   3   2   1   4   2   3   4   2   4   1
 49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
  1   2   3   3   3   2   2   1   3   1   3   2   3   1   3   3   3   2   3   2   1   3   1   4
 73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96
  1   4   3   4   1   1   1   2   1   4   3   4   2   1   3   3   3   2   3   1   2   3   3   3
 97  98  99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
  4   3   1   3   2   3   4   3   2   1   2   3   2   4   1   2   1   2   3   1   1   1   2   2
121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
  3   1   2   3   3   3   3   3   2   4   1   1   3   3   2   1   2   3   3   4   1   1   3   2
145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168
  3   3   2   2   3   3   3   2   2   4   3   3   1   2   3   3   1   3   4   2   2   2   2   3
169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192
  2   3   1   2   1   1   1   1   1   4   3   2   3   3   1   3   3   3   2   4   3   3   3   2
193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216
  2   2   2   1   3   3   3   3   1   2   3   1   1   3   2   3   3   4   2   1   3   1   3   3
217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
  1   3   3   4   4   3   4   1   4   1   3   1   2   3   1   4   3   1   3   3   4   3   3   3
241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264
  1   2   3   1   1   4   3   1   1   3   1   1   4   3   2   3   4   2   3   3   4   1   3   1
265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288
  1   2   1   3   2   2   3   2   3   4   3   2   3   3   3   1   3   3   2   4   3   3   3   2
289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312
  4   4   3   4   3   1   1   1   3   3   3   4   1   2   3   1   2   3   1   2   3   2   1   1
313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336
  2   3   3   3   3   1   3   2   2   1   1   2   3   1   1   1   4   3   2   2   4   1   3   3
337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
  2   4   3   2   3   1   3   2   2   3   3   3   1   3   1   3   3   1   3   1   3   3   1   2
361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384
  1   3   3   1   2   3   1   3   2   3   1   4   4   4   3   3   3   2   2   2   2   3   1   3
385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408
  3   3   3   1   3   2   4   1   3   1   3   1   2   3   3   3   3   3   3   3   1   1   3   2
409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432
  1   3   1   3   2   1   1   2   3   1   3   3   4   3   3   3   3   3   1   3   2   1   4   3
433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456
  1   3   3   3   2   3   2   1   4   3   3   2   3   3   3   2   1   3   3   3   3   3   4   3
457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480
  2   3   1   2   1   2   4   1   3   4   1   3   3   3   3   2   1   1   3   2   2   1   4   3
481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504
  3   3   1   2   4   3   2   3   1   2   3   3   3   4   2   1   3   3   1   3   3   1   4   2
505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528
  1   1   1   3   4   3   2   3   3   2   2   3   4   1   2   1   3   3   2   3   1   2   2   3
529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552
  1   3   1   1   1   2   1   3   4   1   2   2   2   4   1   3   1   3   4   1   3   2   3   3
553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576
  3   4   3   3   2   3   3   3   1   3   3   3   4   3   2   3   3   3   2   1   1   1   1   1
577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600
  1   1   2   3   1   1   3   1   2   1   4   3   4   4   2   3   2   3   2   4   3   1   4   3
601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624
  1   2   4   3   2   3   3   3   1   4   3   3   3   3   2   2   1   4   3   2   2   1   3   2
625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648
  1   2   4   3   2   2   1   1   3   2   3   1   3   2   1   1   1   3   3   1   4   3   2   1
649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672
  3   2   1   1   1   4   3   4   4   3   3   4   2   3   1   3   1   3   2   3   3   1   3   3
673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696
  2   1   3   2   3   3   3   2   1   3   2   1   2   4   1   1   3   3   3   3   3   3   3   3
697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720
  1   2   2   3   3   3   1   1   2   3   2   3   3   2   1   3   2   1   3   3   1   1   3   1
721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744
  2   3   3   3   2   2   3   1   4   3   3   1   3   1   4   3   4   3   3   2   2   3   3   3
745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768
  3   1   2   1   3   3   2   1   3   4   1   3   2   3   4   3   4   1   1   2   4   3   2   3
769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792
  4   2   1   2   3   3   3   3   2   3   2   3   3   3   2   1   3   3   2   3   3   2   4   3
793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816
  3   3   2   3   3   1   2   3   4   3   1   2   3   4   3   3   1   2   3   1   3   2   3   1
817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840
  1   3   2   1   4   1   2   1   1   3   3   3   3   1   1   3   3   2   3   1   3   1   4   2

  841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864
    3   1   3   1   3   4   3   3   3   4   2   4   3   3   4   3   4   4   2   1   3   4   3   3
  865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888
    3   3   2   1   2   2   3   3   3   3   1   2   3   3   3   2   3   2   3   1   3   2   2   2
  889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912
    1   3   2   2   2   1   2   2   3   4   3   3   3   3   3   3   3   3   1   3   2   3   2   3
  913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936
    3   1   2   3   2   4   1   2   3   3   2   1   4   2   3   1   4   4   3   2   4   2   3   1
  937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960
    2   4   2   4   1   3   3   3   2   3   1   1   2   3   3   3   4   3   2   2   3   3   3   1
  961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984
    4   3   3   1   1   2   1   3   4   3   3   3   3   1   2   3   3   3   4   1   3   2   1
  985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000
    4   3   1   2   2   3   3   3   1   1   1   3   3   1   4
  [ reached getOption("max.print") -- omitted 201 entries ]

Within cluster sum of squares by cluster:
[1]  682.5247  684.7234 1692.4575  580.4621
 (between_SS / total_SS =  49.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

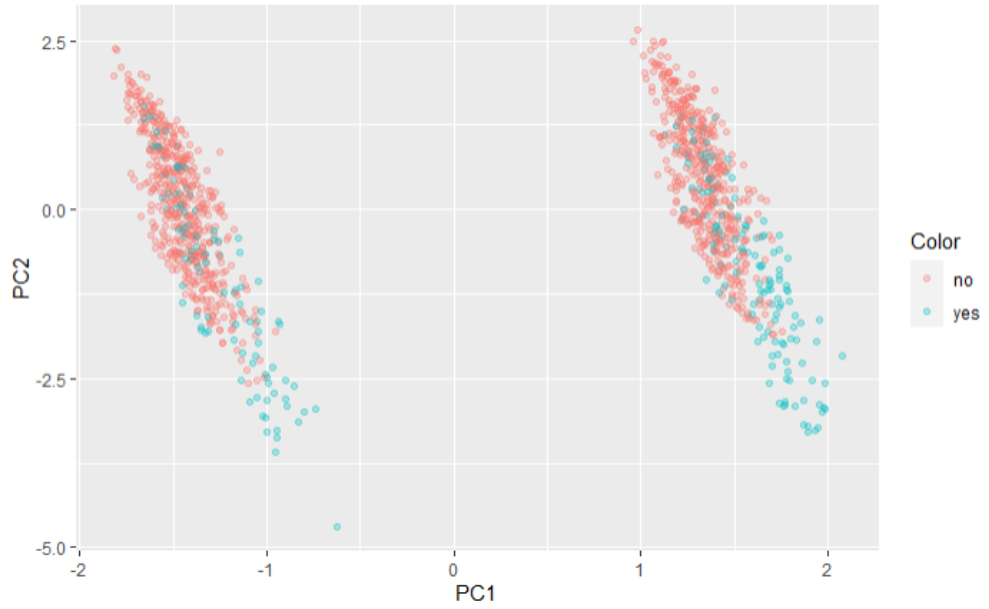
Based on this, we can visualize how the clusters were formed:

```
#Display the cluster plot
fviz_cluster(fit, data = predictors)
```



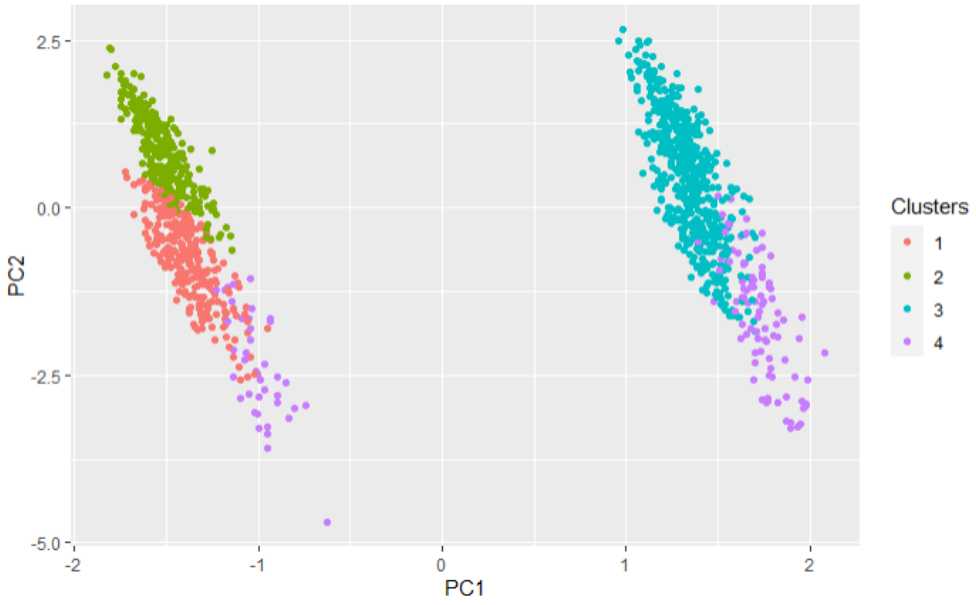
For comparison we can generate our own PCA plot and colour the points based on smokers.

```
#Calculate PCA
pca = prcomp(predictors)
#Save as dataframe
rotated_data = as.data.frame(pca$x)
#Add original labels as a reference
rotated_data$Color <- insurance$smoker
#Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) + geom_point(alpha = 0.3)
```



The cluster plot can also be done on ggplot based on the cluster result from the K-means algorithm:

```
#Assign the clusters as a new column
rotated_data$Clusters = as.factor(fit$cluster)
#Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Clusters)) + geom_point()
```

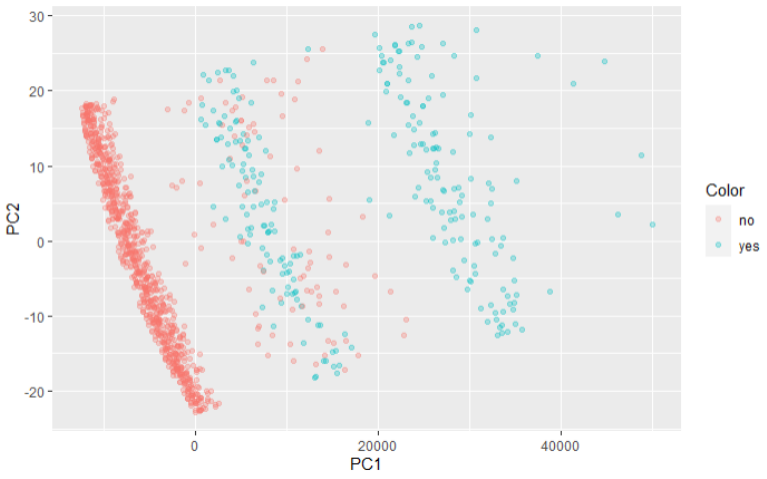


f. Classification

Use at least two classifiers to predict a label in your data. If a label was not provided with the data, use the clustering from the previous part. Follow the process for choosing the best parameters for your choice of classifier. Compare the accuracy of the two.

First, I used PCA and dummy variables to create a projection of the data to 2D and to show a scatterplot with color showing 'yes' (or) 'no' for smoking.

```
set.seed(123)
#Calculate PCA
pca = prcomp(dummies)
#Save as dataframe
rotated_data = as.data.frame(pca$x)
#Add original label 'smoker' as a reference
rotated_data$Color <- insurance$smoker
#Plot and color the labels based on 'yes' (or) 'no' for smoking
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) + geom_point(alpha = 0.3)
```



We can see that smoking may be a good decision-maker in insurance. Now, I will use the kNN, decision trees, and SVM to predict the smokers from the rest of the variables.

KNN:

```
#KNN
set.seed(123)
ctrl <- trainControl(method="cv", number = 10)
knnFit <- train(smoker ~ ., data = insurance,
                method = "knn",
                trControl = ctrl,
                preProcess = c("center", "scale"),
                tuneLength = 15)
knnFit
```

```
k-Nearest Neighbors

1201 samples
   6 predictor
   2 classes: 'no', 'yes'

Pre-processing: centered (8), scaled (8)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1082, 1081, 1081, 1082, 1080, 1082, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9341219  0.7916194
   7  0.9200170  0.7368963
   9  0.9158014  0.7167635
  11  0.9050230  0.6656495
  13  0.9075163  0.6688325
  15  0.9025298  0.6478936
  17  0.9050369  0.6556338
  19  0.9033979  0.6474017
  21  0.9042383  0.6487636
  23  0.9050856  0.6496547
  25  0.9075788  0.6564748
  27  0.9067385  0.6521208
  29  0.9059052  0.6487935
  31  0.9050648  0.6450638
  33  0.9050648  0.6450638

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 5.
```

For K=5:

```
fit <- kmeans(dummies, centers = 5, nstart = 25)
#Display the kmeans object information
fit
```

```
K-means clustering with 5 clusters of sizes 158, 420, 89, 57, 477

Cluster means:
      age sexfemale    sexmale      bmi children   charges
1 41.82278 0.4873418 0.5126582 28.01187 1.208861 22379.800
2 52.12143 0.5404762 0.4595238 30.96919 1.154762 11317.146
3 37.30337 0.3258427 0.6741573 33.29107 1.280899 37476.293
4 51.28070 0.4210526 0.5789474 37.26009 1.245614 47388.517
5 31.83019 0.5010482 0.4959518 30.19289 1.140461  4776.254

Clustering vector:
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23
   5    1    5    5    2    5    5    1    5    1    5    2    3    2    5    2    3    2    5    3    5    2    2
  24   25   26   27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46
   2    5    3    3    2    4    2    5    3    4    5    5    5    5    5    1    5    2    3    5    1    3    2
  47   48   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64   65   66   67   68   69
   4    2    1    5    2    5    3    5    2    2    5    5    1    2    5    2    2    5    2    5    5    5    5
  70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85   86   87   88   89   90   91   92
   5    5    3    2    3    1    4    2    5    2    5    2    3    5    4    5    2    2    1    2    5    5    3
  93   94   95   96   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115
   5    1    5    5    4    2    2    2    5    5    2    3    2    1    2    5    5    3    2    5    2    3    5
 116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138
   2    2    2    5    5    5    1    5    1    5    1    1    1    5    3    2    2    5    5    5    1    5    5
 139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161
   1    3    1    1    2    5    5    2    5    5    5    2    2    5    5    4    5    2    2    2    2    5    5
 162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184
   5    4    5    5    5    5    2    5    5    2    5    2    2    2    2    2    3    5    5    2    1    2    5
 185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207
   5    2    5    5    2    5    2    5    5    1    5    2    5    1    2    5    1    5    2    5    2    2    5
 208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230
   1    5    3    3    3    3    5    1    1    2    5    5    4    4    5    3    2    4    5    2    2    1    1
 231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253
   1    4    1    2    5    2    4    5    2    5    2    5    2    2    1    4    5    2    2    5    2    2    4
 254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276
   1    5    1    3    3    5    1    3    2    5    1    2    5    2    1    1    5    2    5    2    3    2    3
 277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299
   2    2    2    5    5    5    1    3    2    5    5    5    4    4    2    4    1    2    2    1    2    2    2
 300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322
   3    2    1    2    2    2    2    5    5    2    5    2    2    5    2    1    2    2    5    2    5    5    5
 323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345
   2    2    5    2    2    2    5    3    5    1    1    3    2    1    2    3    1    5    2    2    3    5    5
 346  347  348  349  350  351  352  353  354  355  356  357  358  359  360  361  362  363  364  365  366  367  368
   2    2    2    2    5    2    2    2    2    2    2    2    2    5    2    5    2    5    5    1    2    5    2
 369  370  371  372  373  374  375  376  377  378  379  380  381  382  383  384  385  386  387  388  389  390  391
   1    2    1    4    4    3    5    2    2    5    5    1    5    5    2    5    2    5    5    2    5    5    3
 392  393  394  395  396  397  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413  414
   1    1    5    2    2    5    5    5    5    5    5    5    1    2    2    2    5    2    1    2    2    1    2
 415  416  417  418  419  420  421  422  423  424  425  426  427  428  429  430  431  432  433  434  435  436  437
   2    1    5    1    1    1    3    5    5    5    2    2    2    2    5    2    4    2    1    2    1    5    5
 438  439  440  441  442  443  444  445  446  447  448  449  450  451  452  453  454  455  456  457  458  459  460
   5    2    2    3    5    1    5    5    5    5    2    2    5    1    5    2    1    2    1    5    5    1    5
 461  462  463  464  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480  481  482  483
   2    5    3    2    2    4    2    2    1    2    5    5    2    2    1    5    5    2    4    2    1    5    2
 484  485  486  487  488  489  490  491  492  493  494  495  496  497  498  499  500  501  502  503  504  505  506
   5    4    2    5    2    2    1    5    2    5    3    2    5    2    5    3    2    5    5    2    4    3    2
 507  508  509  510  511  512  513  514  515  516  517  518  519  520  521  522  523  524  525  526  527  528  529
   2    5    4    2    5    2    5    1    5    5    4    2    5    2    2    5    2    5    2    5    5    5    3
 530  531  532  533  534  535  536  537  538  539  540  541  542  543  544  545  546  547  548  549  550  551  552
   2    2    2    5    1    5    3    2    5    5    5    4    2    1    2    5    3    2    2    5    5    2
 553  554  555  556  557  558  559  560  561  562  563  564  565  566  567  568  569  570  571  572  573  574  575
   2    4    2    5    5    5    2    2    1    1    2    5    3    2    5    1    2    5    2    2    2    2    2
 576  577  578  579  580  581  582  583  584  585  586  587  588  589  590  591  592  593  594  595  596  597  598
   2    2    1    5    5    1    2    5    1    5    1    4    2    3    4    5    5    5    5    5    4    5    2
 599  600  601  602  603  604  605  606  607  608  609  610  611  612  613  614  615  616  617  618  619  620  621
   4    2    2    5    3    2    5    2    5    5    1    3    5    2    5    5    5    5    1    3    2    5    5
 622  623  624  625  626  627  628  629  630  631  632  633  634  635  636  637  638  639  640  641  642  643  644
   2    2    5    2    5    4    2    5    5    2    2    5    5    2    2    2    2    2    2    2    2    2    3
 645  646  647  648  649  650  651  652  653  654  655  656  657  658  659  660  661  662  663  664  665  666  667
   5    2    5    1    2    5    2    2    2    3    5    3    4    2    1    4    5    2    2    2    2    5    1
 668  669  670  671  672  673  674  675  676  677  678  679  680  681  682  683  684  685  686  687  688  689  690
   5    2    2    1    5    5    1    5    5    5    1    5    2    2    2    5    2    5    1    2    2    5    2
 691  692  693  694  695  696  697  698  699  700  701  702  703  704  705  706  707  708  709  710  711  712  713
   5    5    5    2    1    2    1    5    5    2    5    5    2    5    1    5    1    5    5    2    1    5
 714  715  716  717  718  719  720  721  722  723  724  725  726  727  728  729  730  731  732  733  734  735  736
   2    5    5    5    1    5    2    5    2    5    5    5    5    1    4    5    5    2    2    2    4    1
 737  738  739  740  741  742  743  744  745  746  747  748  749  750  751  752  753  754  755  756  757  758  759
   3    5    2    5    5    2    5    5    5    2    5    2    5    2    3    1    2    4    2    5    5    2    3

 760  761  762  763  764  765  766  767  768  769  770  771  772  773  774  775  776  777  778  779  780  781  782
   2    3    2    1    5    3    2    1    2    4    5    2    5    2    5    2    4    5    2    5    5    5    2
 783  784  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800  801  802  803  804  805
   5    1    5    5    5    5    5    5    4    5    1    1    5    5    2    1    5    2    4    2    2    1    2
 806  807  808  809  810  811  812  813  814  815  816  817  818  819  820  821  822  823  824  825  826  827  828
   4    5    2    2    5    5    3    2    1    5    2    5    5    5    1    3    2    3    2    2    5    5    5
 829  830  831  832  833  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849  850  851
   1    2    2    5    5    5    2    5    5    2    3    2    2    2    2    2    5    3    5    1    2    4    5
 852  853  854  855  856  857  858  859  860  861  862  863  864  865  866  867  868  869  870  871  872  873  874
   3    1    5    3    2    3    1    5    2    2    1    5    1    5    5    2    2    5    5    5    2    5    5
 875  876  877  878  879  880  881  882  883  884  885  886  887  888  889  890  891  892  893  894  895  896  897
   2    5    1    5    1    2    5    5    2    1    5    2    5    5    2    5    2    5    5    2    5    5    1
 898  899  900  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915  916  917  918  919  920
   3    5    1    2    5    5    1    1    2    2    1    3    2    5    2    3    2    1    2    3    3    2    5
 921  922  923  924  925  926  927  928  929  930  931  932  933  934  935  936  937  938  939  940  941  942  943
   1    2    5    1    4    5    2    2    3    3    5    1    3    5    2    1    5    4    5    3    5    2    2
 944  945  946  947  948  949  950  951  952  953  954  955  956  957  958  959  960  961  962  963  964  965  966
   1    5    2    2    1    5    5    5    2    4    5    5    5    2    5    2    2    2    2    5    2    2    5
 967  968  969  970  971  972  973  974  975  976  977  978  979  980  981  982  983  984  985  986  987  988  989
   2    5    3    2    5    5    5    2    1    2    2    2    2    3    2    5    3    2    4    5    1    5    2
 990  991  992  993  994  995  996  997  998  999 1000
   2    5    2    1    2    2    5    2    2    3
 [ reached getOption("max.print") -- omitted 201 entries ]

Within cluster sum of squares by cluster:
[1] 1873147993 1908999781  957818767 1186799633 1547477966
 (between_SS / total_SS =  95.7 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

SVM:

```
#SVM
#I decided to usde the grid search here to try different values of C
grid <- expand.grid(C = 10^seq(-5,2,0.5))
train_control = trainControl(method = "cv", number = 10)
#Fit the model
svm_grid <- train(smoker ~ ., data = insurance, method = "svmLinear",
                  trControl = train_control, tuneGrid = grid)
#View grid search result
svm_grid
```

```
Support Vector Machines with Linear Kernel

1201 samples
   6 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1080, 1081, 1081, 1080, 1080, 1082, ...
Resampling results across tuning parameters:

  C            Accuracy   Kappa
  1.000000e-05  0.7968471  0.0000000
  3.162278e-05  0.7968471  0.0000000
  1.000000e-04  0.7968471  0.0000000
  3.162278e-04  0.7968471  0.0000000
  1.000000e-03  0.8117782  0.1037204
  3.162278e-03  0.8901009  0.6047026
  1.000000e-02  0.8884411  0.6029638
  3.162278e-02  0.9075876  0.6867888
  1.000000e-01  0.9667560  0.9044001
  3.162278e-01  0.9675754  0.9066960
  1.000000e+00  0.9675823  0.9066883
  3.162278e+00  0.9700755  0.9134782
  1.000000e+01  0.9700755  0.9134782
  3.162278e+01  0.9700755  0.9134782
  1.000000e+02  0.9700755  0.9134782

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 3.162278.
```

Decision Trees:

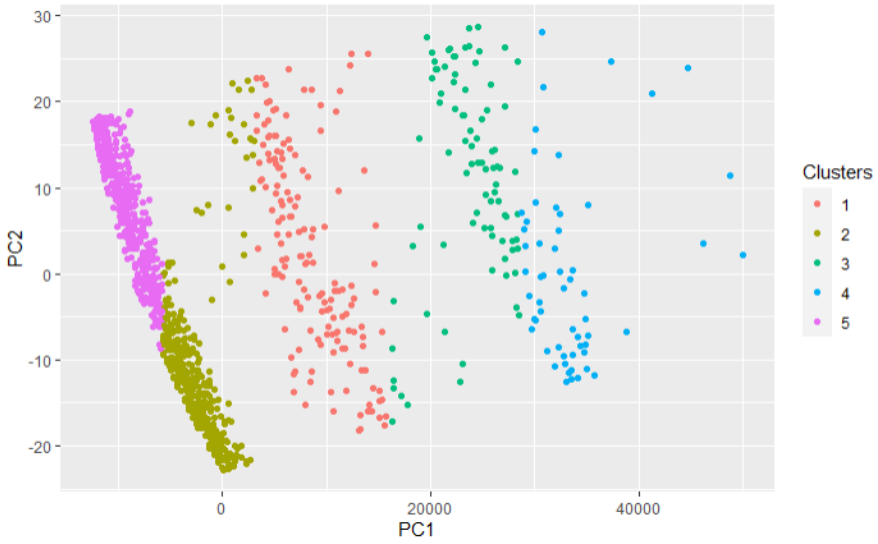
```
#Decision Trees
#Evaluation Method
train_control = trainControl(method = "cv", number = 10)
#Fit the Model
tree1 <- train(smoker ~ ., data = insurance, method = "rpart", trControl = train_control)
#Evaluate the fit
tree1
```

```
CART

1201 samples
   6 predictor
   2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1081, 1081, 1081, 1081, 1081, 1081, 1081, ...
Resampling results across tuning parameters:

  cp           Accuracy   Kappa
  0.008196721  0.9583672  0.8722621
  0.092213115  0.9383456  0.8273615
  0.651639344  0.8484145  0.3666238

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.008196721.
```

Accuracy seems to be comparable in decision trees and SVM, although SVM is consistently more accurate (96 %+).

Now, we can visualize the labels for KNN as we did previously with PCA:

```
#Assign clusters as a new column
rotated_data$Clusters = as.factor(fit$cluster)
#Plot and color by labels
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Clusters)) + geom_point()
```



As a reminder, here is the visualization with PCA labels:

```
#Plot and color the labels based on wine type red or white
ggplot(data = rotated_data, aes(x = PC1, y = PC2, col = Color)) + geom_point(alpha = 0.3)
```

PCA seemed to depict the clusters into two logical clusters which compares to the charges distribution based on your smoking habit. The KNN clustering method provided an interesting distribution, although I prefer PCA as it seems to make more sense compared to our exploratory work.

g. Evaluation

Using the better classifier from the previous step, perform a more sophisticated evaluation using the tools of Week 9. Specifically, (1) produce a 2x2 confusion

matrix (if your dataset has more than two classes, bin the classes into two groups and rebuild the model), (2) calculate the precision and recall manually, and finally (3) produce an ROC plot (see Tutorial 9). Explain how these performance measures makes your classifier look compared to accuracy.

So I was supposed to use this step to predict whether the person is a smoker (or) not, but I had some issues trying to run the model with this dataset. Therefore, I decided to, for the sake of practice, determine the age based on age groups. I will create a new column by binning ages into groups and creating a new column for them. While binning the classes into 2 groups is what the question is asking for, this didn't work with my dataset. Therefore, I have decided to create 5 bins instead based on age groups, which is more logical for the sake of this example.

First, let's create new bins for those age groups from 20 to 60+:

```
#Create a new file for the modified dataset
myinsurance <- insurance
#Remove regions
myinsurance <- myinsurance %>% select(-c(region))
#Bin age groups into 5 different bins
myinsurance <- myinsurance %>% mutate(agegroup = cut(age,breaks=c(-Inf, 29, 39, 49, 59, Inf), labels=c("twenties", "thirties",
"fourties", "fifties", "sixtieplus")))
#Now remove age
myinsurance <- myinsurance %>% select(-c(age))
head(myinsurance)
```

A tibble: 6 x 6

sex <chr>	bmi <dbl>	children <dbl>	smoker <chr>	charges <dbl>	agegroup <fctr>
male	33.000	3	no	4449.462	twenties
male	22.705	0	no	21984.471	thirties
male	28.880	0	no	3866.855	thirties
female	25.740	0	no	3756.622	thirties
female	33.440	1	no	8240.590	fourties
female	27.740	3	no	7281.506	thirties

6 rows

Let's create a 70-30 train test split:

```
#Set seed
set.seed(123)
#Partition the data
index = createDataPartition(y=myinsurance$agegroup, p=0.7, list = FALSE)
#Everything in the generated index list
train_set = myinsurance[index,]
#Everything except the generated indices
test_set = myinsurance[-index,]
```

Now let's fit the model:

```
#Fit the model using the training set
svm_split <- train(agegroup ~ ., data = train_set, method = "svmLinear")
#Predict with test set
pred_split <- predict(svm_split, test_set)
train_control_boot = trainControl(method = "boot", number = 100)
#Fit the model
svm <- train(agegroup ~ ., data = myinsurance, method = "svmLinear",
             trControl = train_control_boot)
#Evaluate Fit
svm
```

```
Support Vector Machines with Linear Kernel

1201 samples
   5 predictor
   5 classes: 'twenties', 'thirties', 'fourties', 'fifties', 'sixtieplus'

No pre-processing
Resampling: Bootstrapped (100 reps)
Summary of sample sizes: 1201, 1201, 1201, 1201, 1201, 1201, ...
Resampling results:

  Accuracy   Kappa
  0.4372235  0.2738834

Tuning parameter 'C' was held constant at a value of 1
```

Now, we can generate a confusion matrix:

```
#Generate confusion matrix for the test set
cm <- confusionMatrix(test_set$agegroup, pred_split)
cm
```

```
Confusion Matrix and Statistics

           Reference
Prediction   twenties thirties fourties fifties sixtieplus
  twenties         60       15        5       4          0
  thirties         40       24        7       6          0
  fourties         19       29       21      12          2
  fifties           2        4       22      45          8
  sixtieplus        1        1        5      23          4

Overall Statistics

               Accuracy : 0.429
                 95% CI : (0.3772, 0.482)
    No Information Rate : 0.3398
    P-Value [Acc > NIR] : 0.0002759

                  Kappa : 0.266

 Mcnemar's Test P-Value : 7.573e-07

Statistics by Class:

                     Class: twenties Class: thirties Class: fourties Class: fifties Class: sixtieplus
Sensitivity                   0.4918          0.32877          0.3500          0.5000           0.28571
Specificity                   0.8987          0.81469          0.7926          0.8662           0.91304
Pos Pred Value                0.7143          0.31169          0.2530          0.5556           0.11765
Neg Pred Value                0.7745          0.82624          0.8587          0.8381           0.96923
Prevalence                    0.3398          0.20334          0.1671          0.2507           0.03900
Detection Rate                0.1671          0.06685          0.0585          0.1253           0.01114
Detection Prevalence          0.2340          0.21448          0.2312          0.2256           0.09471
Balanced Accuracy             0.6953          0.57173          0.5713          0.6831           0.59938
```

We can view the scoring metrics by class:

```
#Store the byClass object of confusion matrix as a dataframe
metrics <- as.data.frame(cm$byClass)
#View the object
metrics
```

Description: df [5 x 11]

	Sensitivity <dbl>	Specificity <dbl>	Pos Pred Value <dbl>	Neg Pred Value <dbl>	Precision <dbl>	Recall <dbl>	F1 <dbl>
Class: twenties	0.4918033	0.8987342	0.7142857	0.7745455	0.7142857	0.4918033	0.5825243
Class: thirties	0.3287671	0.8146853	0.3116883	0.8262411	0.3116883	0.3287671	0.3200000
Class: fourties	0.3500000	0.7926421	0.2530120	0.8586957	0.2530120	0.3500000	0.2937063
Class: fifties	0.5000000	0.8661710	0.5555556	0.8381295	0.5555556	0.5000000	0.5263158
Class: sixtieplus	0.2857143	0.9130435	0.1176471	0.9692308	0.1176471	0.2857143	0.1666667

5 rows | 1-8 of 11 columns

Precision:

```
#Get the precision value for each class
metrics %>% select(c(Precision))
```

Description: df [5 x 1]

	Precision <dbl>
Class: twenties	0.7142857
Class: thirties	0.3116883
Class: fourties	0.2530120
Class: fifties	0.5555556
Class: sixtieplus	0.1176471

5 rows

Recall:

```
#Get the recall value for each class
metrics %>% select(c(Recall))
```

Description: df [5 x 1]

	Recall <dbl>
Class: twenties	0.4918033
Class: thirties	0.3287671
Class: fourties	0.3500000
Class: fifties	0.5000000
Class: sixtieplus	0.2857143

5 rows

Specifity:

```
#Get the specificity value for each class
metrics %>% select(c(Specificity))
```

Description: df [5 x 1]

	Recall <dbl>
Class: twenties	0.4918033
Class: thirties	0.3287671
Class: fourties	0.3500000
Class: fifties	0.5000000
Class: sixtieplus	0.2857143

5 rows

F1 Score:

```
#Get the F1 score value for each class
metrics %>% select(c(F1))
```

Description: df [5 x 1]

	F1 <dbl>
Class: twenties	0.5825243
Class: thirties	0.3200000
Class: fourties	0.2937063
Class: fifties	0.5263158
Class: sixtieplus	0.1666667

5 rows

And Balanced Accuracy:

```
#Get the balanced accuracy value for each class
metrics %>% select(c('Balanced Accuracy'))
```

Description: df [5 x 1]

	Balanced Accuracy <dbl>
Class: twenties	0.6952687
Class: thirties	0.5717262
Class: fourties	0.5713211
Class: fifties	0.6830855
Class: sixtieplus	0.5993789

5 rows

Now, since we want to produce ROC plot, I will need to go back to the original class we are predicting, which is smokers.

```
insurance$smoker <- as.factor(insurance$smoker)
#Partition the data
index = createDataPartition(y=insurance$smoker, p=0.7, list = FALSE)
#Everything in the generated index list
train_ins = insurance[index,]
#Everything except the generated indices
test_ins = insurance[-index,]
```

Now building a KNN model with cross validation:

```
#Set control parameter
train_control = trainControl(method = "cv", number = 10)
#Fit the Model
knn <- train(smoker ~., data = train_ins, method = "knn", trControl = train_control, tuneLength = 20)
#Evaluate Fit
knn
```

```
k-Nearest Neighbors

841 samples
  6 predictor
  2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 757, 757, 757, 757, 757, 757, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9167647  0.7501966
   7  0.9227031  0.7663498
   9  0.9286275  0.7894054
  11  0.9250840  0.7760368
  13  0.9298459  0.7929790
  15  0.9334034  0.8044700
  17  0.9334034  0.8054280
  19  0.9322129  0.8021138
  21  0.9334034  0.8056202
  23  0.9345798  0.8094370
  25  0.9369608  0.8159630
  27  0.9334034  0.8056962
  29  0.9298459  0.7944797
  31  0.9286555  0.7903813
  33  0.9286555  0.7895893
  35  0.9334034  0.8028013
  37  0.9274790  0.7851321
  39  0.9322409  0.7990821
  41  0.9322269  0.8015185
  43  0.9286555  0.7905287

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 25.
```

Now, generate a confusion matrix:

```
#Evaluate the fit with a confusion matrix
pred_ins <- predict(knn, test_ins)
#Confusion Matrix
confusionMatrix(test_ins$smoker, pred_ins)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  274  13
       yes  10  63

               Accuracy : 0.9361
                 95% CI : (0.9057, 0.9591)
    No Information Rate : 0.7889
    P-Value [Acc > NIR] : 8.994e-15

                  Kappa : 0.8054

 Mcnemar's Test P-Value : 0.6767

            Sensitivity : 0.9648
            Specificity : 0.8289
         Pos Pred Value : 0.9547
         Neg Pred Value : 0.8630
             Prevalence : 0.7889
         Detection Rate : 0.7611
   Detection Prevalence : 0.7972
      Balanced Accuracy : 0.8969

       'Positive' Class : no
```

Now, we can generate an ROC table for our data:

```
library(pROC)
#Get class probabilities for KNN
pred_prob <- predict(knn, test_ins, type = "prob")
head(pred_prob)
```

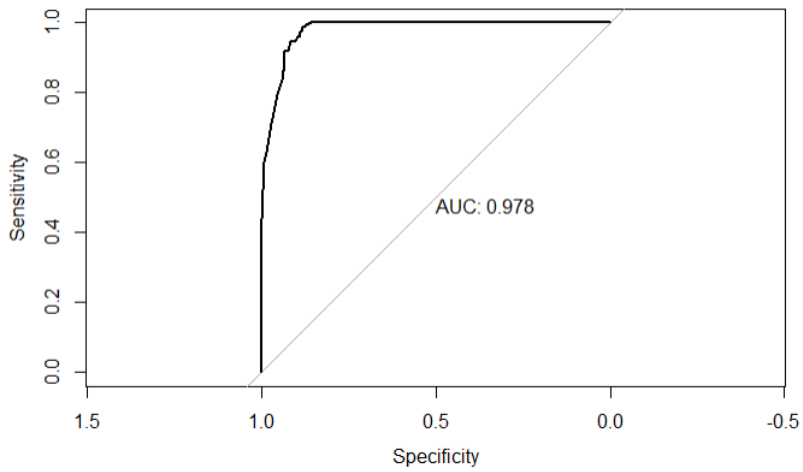
Description: df [6 x 2]

	no <dbl>	yes <dbl>
1	1.0000000	0.0000000
2	1.0000000	0.0000000
3	1.0000000	0.0000000
4	1.0000000	0.0000000
5	1.0000000	0.0000000
6	0.6666667	0.3333333

6 rows

And now, an ROC curve:

```
#And now we can create an ROC curve for our model.
roc_obj <- roc((test_ins$smoker), pred_prob[,1])
plot(roc_obj, print.auc=TRUE)
```



Based on our findings, the 'no' class was definitely favoured since most of those in the dataset are non-smokers. While the AUC metric is supposed to combine sensitivity and specificity to give us a good middle ground metric, this may be misleading for how imbalanced the number of smokers are.

h. <u>Report</u>

In a single document, include the answers to all of the parts of this Problem, including this one. The report component specifically is about your overall takeaways from your data. What was interesting from your analysis?

So to summarize:

- There was a clear imbalance in the number of smokers versus the number if non-smokers in the dataset, which made for a challenging clustering task.
- Smoking could be an indicator of insurance costs increasing. That is, there is positive correlation between smoking and the increase of insurance charges, which was part of our hypothesis.
- The non-smokers were rare in each age group, and there was no true correlation between age and smoking habits.
- Finally, our model struggled to predict age groups based on the data provided with about 50% accuracy reported, on average.

i. Reflection

The final section of the report is a (short) paragraph reflecting on the course as a whole and what you have learned. The goal is not actually feedback for the course but to get you to think back about what you have learned and how your perspective on data science has changed.

I think one way that this course changed my perspective on data science is by showing me how, regardless of the data, there is a correlation that could be drawn somewhere with some head scratching and analysis. I now always have a pending hypothesis to set a tone for each project as it can provide me with some guidance. If there is no clear correlation, it is important to perform exploratory tasks to view possible correlations that may influence (or) create a new hypothesis. Based on that, I can run my analysis and modelling, then reach conclusions to prove (or) disapprove my pending hypothesis. Overall, this course taught me the 'scientific method' that we learned about in elementary from a data science perspective.