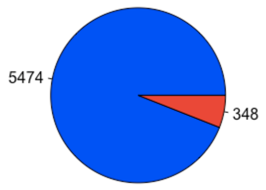
Farhana Afroze
UB Person# 50158114

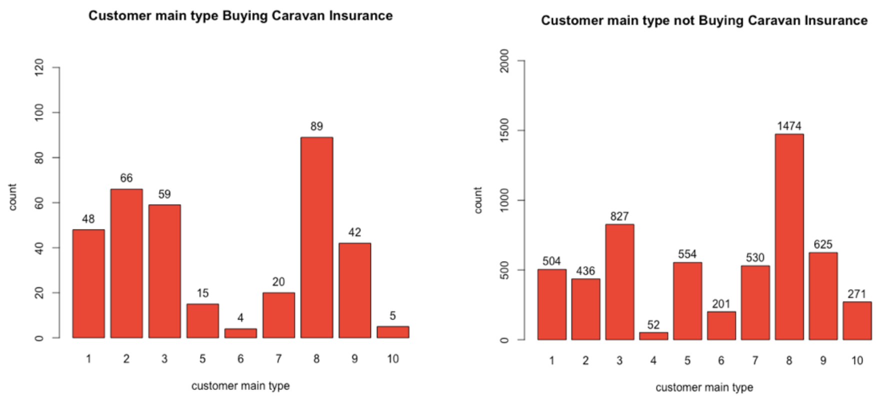
1.

Exploring dataset and visualization:

At first, we looked at the train data and test data which was given in the dataset. In test data there is no response variable called caravan_insurace. In the train data with response variable there is 5822 rows and 86 columns. There are no missing values in the dataset. In the response variable caravan_insurance there is two categorical value 0 and 1. So, 0 represents customer not going to buy caravan insurance and 1 represents customers going to buy caravan insurance. So in the train dataset there is 5474 0's and 348 1's.

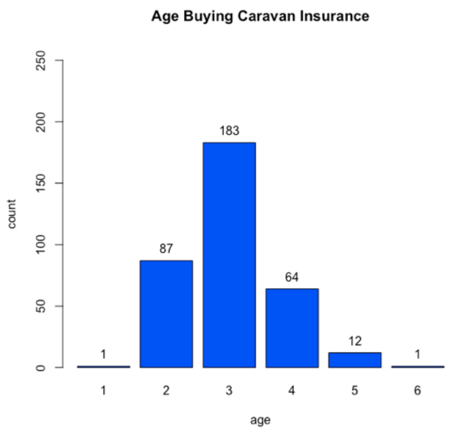


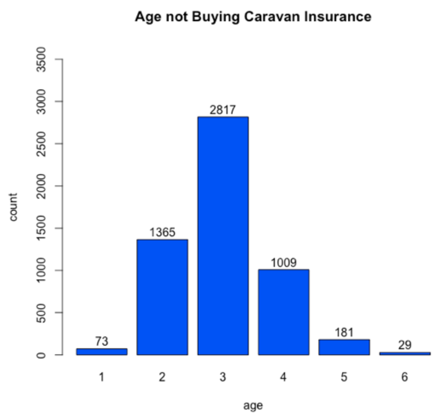
There is features called main type customer in the train set and it has 10 types of customers such as successful hedonists, career loners, cruising seniors and others type. We would like to see who buy caravan insurance and who don't. Here is the plot who buy caravan insurance.



Here we see 8 is higher than any other types of customer. So, family with grown- ups buying 89 caravan insurance. And we see type 4 is not exist so career loners are not buying caravan insurance. In the right side plot we see 1474 grown-ups aren't buying insurance either. So basically, out of 1500 grown-ups only 89 are buying caravan insurance.

Next we would like to see age of the customer who are buying caravan insurance and also who aren't buying the insurance. Here are both plot:

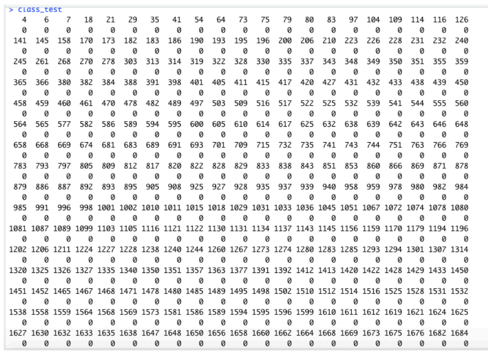




Here we have 6 age groups. Here we see most people who are buying caravan insurance is age group 3 and their age is between 40-50 years. So total 3000 people age between 40 to 50 not buying caravan insurance is 2817.

Data preprocessing:
First, we would like to split the train dataset into training and testing as in the main testing set, we don't have the response variable we need for our model. After splitting 80% for the train and 20% for the test we have 5822 rows with 86 columns in training and 1165 rows with 86 columns in testing.

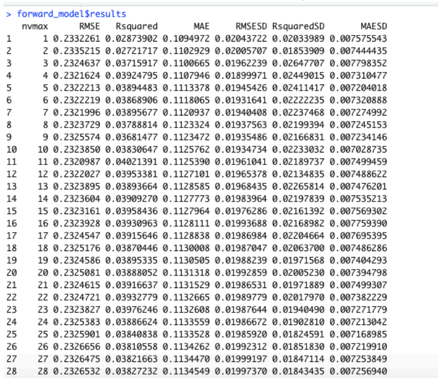
Fitting liner model in training and testing data:
So, after fitting linear model we found out that training MSE is- 242.384 and RMSE is 15.568. Here MSE error is very high because this dataset is not good for OLS as it has categorical variable for response. Next we found out test MSE is – 0.83 which is also very high so this OLS model is not doing good on this data set. Now let's see how much this model could classify the data correctly. So, in test data we have 70 buying caravan insurance. After fitting linear model and predicting we found out there is no 1's in the prediction. So, this OLS model couldn't find out who is buying caravan insurance. Instead it's saying that no one did buy caravan insurance. Here is the snap shot of prediction:


```
> class_test
   4    6    7   18   21   29   35   41   54   64   73   75   79   80   83   97  104  109  114  116  126 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 141  145  158  170  173  182  183  186  190  193  195  196  200  206  210  223  226  228  231  232  240 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 245  261  268  270  278  303  313  314  319  322  328  330  335  337  343  348  349  350  351  355  359 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 365  366  380  382  384  388  391  398  401  405  411  415  417  420  427  431  432  433  438  439  450 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 458  459  460  461  470  478  482  489  497  503  509  516  517  522  525  532  539  541  544  555  560 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 564  565  577  582  586  589  594  595  600  605  610  614  617  625  632  638  639  642  643  646  648 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 658  668  669  674  681  683  689  691  693  701  709  715  732  735  741  743  744  751  763  766  769 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 783  793  797  805  809  812  817  820  822  828  829  833  838  843  851  853  860  866  869  871  878 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 879  886  887  892  893  895  905  908  925  927  928  935  937  939  940  958  959  978  980  982  984 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
 985  991  996  998 1001 1002 1010 1011 1015 1018 1029 1031 1033 1036 1045 1051 1067 1072 1074 1078 1080 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
1081 1087 1089 1099 1103 1105 1116 1121 1122 1130 1131 1134 1137 1143 1145 1156 1159 1170 1179 1194 1196 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
1202 1206 1211 1224 1227 1228 1238 1240 1244 1260 1267 1273 1274 1280 1283 1285 1293 1294 1301 1307 1314 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
1320 1325 1326 1327 1335 1340 1350 1351 1357 1363 1377 1391 1392 1412 1413 1420 1422 1428 1429 1433 1450 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
1451 1452 1465 1467 1468 1471 1478 1480 1485 1489 1495 1498 1502 1510 1512 1514 1516 1525 1528 1531 1532 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
1538 1558 1559 1564 1568 1569 1573 1581 1586 1589 1594 1595 1596 1599 1610 1611 1612 1619 1621 1624 1625 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
1627 1630 1632 1633 1635 1638 1647 1648 1650 1656 1658 1660 1662 1664 1668 1669 1673 1675 1676 1682 1684 
   0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0 
```

Here as we can see all 0’s in the prediction. It couldn’t predict train and test data correctly. It just showing all 0’s.

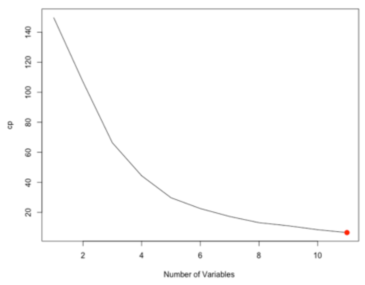
Forward Selection On training data:

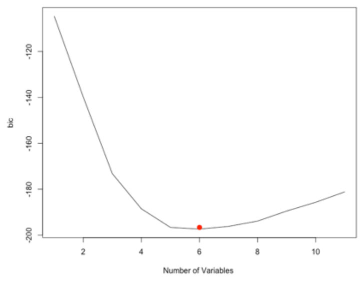
First, we did 10 folds cross validation and then doing forward selection. We are doing each of 50 models and would like to see which model is doing best. After doing forward selection here is the report for each of model.

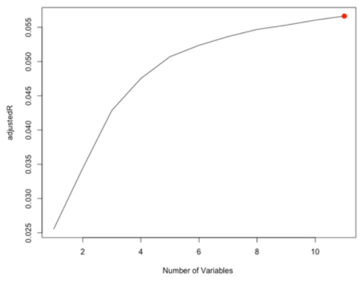


```
> forward_model$results
   nvmax      RMSE   Rsquared       MAE     RMSESD RsquaredSD       MAESD
1      1 0.2332261 0.02873902 0.1094972 0.02043722 0.02033989 0.007575543
2      2 0.2335215 0.02721717 0.1102929 0.02005707 0.01853909 0.007444435
3      3 0.2324637 0.03715917 0.1100665 0.01962239 0.02647707 0.007798352
4      4 0.2321624 0.03924795 0.1107946 0.01899971 0.02449015 0.007310477
5      5 0.2322213 0.03894483 0.1113378 0.01945426 0.02411417 0.007204018
6      6 0.2322219 0.03868906 0.1118065 0.01931641 0.02222235 0.007320888
7      7 0.2321996 0.03895677 0.1120937 0.01940408 0.02237468 0.007274992
8      8 0.2323729 0.03788814 0.1123324 0.01937563 0.02199394 0.007245153
9      9 0.2325574 0.03681477 0.1123472 0.01935486 0.02166831 0.007234146
10    10 0.2323850 0.03830647 0.1125762 0.01934734 0.02233032 0.007028735
11    11 0.2320987 0.04021391 0.1125390 0.01961041 0.02189737 0.007499459
12    12 0.2322027 0.03953381 0.1127101 0.01965378 0.02134835 0.007488622
13    13 0.2323895 0.03893664 0.1128585 0.01968435 0.02265814 0.007476201
14    14 0.2323604 0.03909270 0.1127773 0.01983964 0.02197839 0.007535213
15    15 0.2323161 0.03958436 0.1127964 0.01976286 0.02161392 0.007569302
16    16 0.2323928 0.03930963 0.1128111 0.01993688 0.02168982 0.007759390
17    17 0.2324547 0.03915646 0.1128838 0.01986984 0.02204664 0.007695395
18    18 0.2325176 0.03870446 0.1130008 0.01987047 0.02063700 0.007486286
19    19 0.2324586 0.03895335 0.1130505 0.01988239 0.01971568 0.007404293
20    20 0.2325081 0.03888052 0.1131318 0.01992859 0.02005230 0.007394798
21    21 0.2324615 0.03916637 0.1131529 0.01986531 0.01971889 0.007499307
22    22 0.2324721 0.03932779 0.1132665 0.01989779 0.02017970 0.007382229
23    23 0.2323827 0.03976246 0.1132608 0.01987644 0.01940490 0.007271779
24    24 0.2325383 0.03886624 0.1133559 0.01986672 0.01902810 0.007213042
25    25 0.2325901 0.03840838 0.1133528 0.01985920 0.01824591 0.007168985
26    26 0.2326656 0.03810558 0.1134262 0.01992312 0.01851830 0.007219910
27    27 0.2326475 0.03821663 0.1134470 0.01999197 0.01847114 0.007253849
28    28 0.2326532 0.03827232 0.1134549 0.01997370 0.01843435 0.007256940
```

Here first 28 models are showing. As we see RMSE error for all model is good. But as you see R-squared the 11 model has more variance. 11 model has R-squared 0.04021 and RMSE error is 0.23. So, 11 is the best forward selection model. Next we would like to see cp, bic and adjr2 and would like to see which model these are choosing.



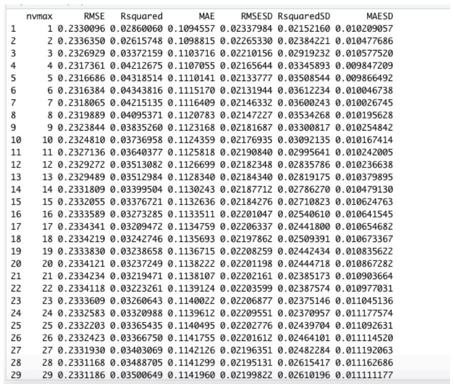




Cp and adjusted-R both choosing model 11 as the best forward selection model. As we can see for case cp it’s going down and at 11 it’s lowest. And in adjusted-R it’s getting up and at 11 model it’s highest. And for bic we see 6 is the best model. Bic penalizes as we include more variables and choose 6 variable model is the best.

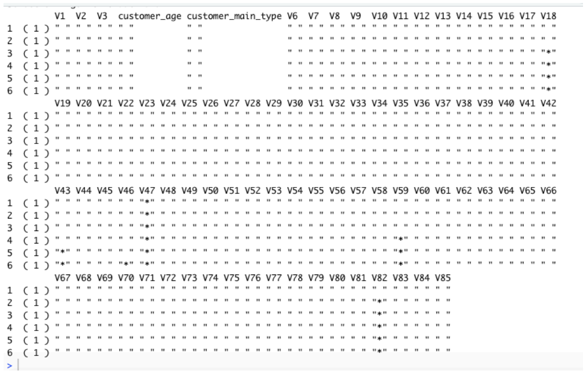
Backward selection on the training data:

Here we also did 10 folds cross validation and did 50 model backward selection to see which model is doing best and chose that model.



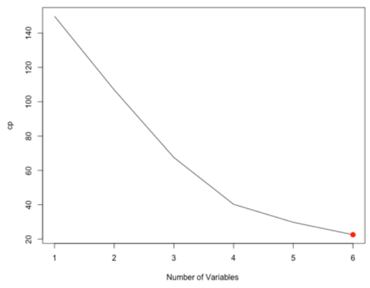
```
   nvmax      RMSE   Rsquared       MAE     RMSESD RsquaredSD       MAESD
1      1 0.2330096 0.02860060 0.1094557 0.02337984 0.02152160 0.010209057
2      2 0.2336350 0.02615748 0.1098815 0.02265330 0.02384221 0.010477686
3      3 0.2326929 0.03372159 0.1103716 0.02210156 0.02919232 0.010577520
4      4 0.2317361 0.04212675 0.1107055 0.02165644 0.03345893 0.009847209
5      5 0.2316686 0.04318514 0.1110141 0.02133777 0.03508544 0.009866492
6      6 0.2316384 0.04343816 0.1115170 0.02131944 0.03612234 0.010046738
7      7 0.2318065 0.04215135 0.1116409 0.02146332 0.03600243 0.010026745
8      8 0.2319889 0.04095371 0.1120783 0.02147227 0.03534268 0.010195628
9      9 0.2323844 0.03835260 0.1123168 0.02181687 0.03300817 0.010254842
10    10 0.2324810 0.03736958 0.1124359 0.02176935 0.03092135 0.010167414
11    11 0.2327136 0.03640377 0.1125818 0.02190840 0.02995641 0.010242005
12    12 0.2329272 0.03513082 0.1126699 0.02182348 0.02835786 0.010236638
13    13 0.2329489 0.03512984 0.1128340 0.02184340 0.02819175 0.010379895
14    14 0.2331809 0.03399504 0.1130243 0.02187712 0.02786270 0.010479130
15    15 0.2332055 0.03376721 0.1132636 0.02184276 0.02710823 0.010624763
16    16 0.2333589 0.03273285 0.1133511 0.02201047 0.02540610 0.010641545
17    17 0.2334341 0.03209472 0.1134759 0.02206337 0.02441800 0.010654682
18    18 0.2334219 0.03242746 0.1135693 0.02197862 0.02509391 0.010673367
19    19 0.2333830 0.03238658 0.1136715 0.02208259 0.02442434 0.010835622
20    20 0.2334121 0.03237249 0.1138222 0.02201198 0.02444718 0.010867282
21    21 0.2334234 0.03219471 0.1138107 0.02202161 0.02385173 0.010903664
22    22 0.2334118 0.03223261 0.1139124 0.02203599 0.02387574 0.010977031
23    23 0.2333609 0.03260643 0.1140022 0.02206877 0.02375146 0.011045136
24    24 0.2332583 0.03320988 0.1139612 0.02209551 0.02370957 0.011177574
25    25 0.2332203 0.03365435 0.1140495 0.02202776 0.02439704 0.011092631
26    26 0.2332423 0.03366750 0.1141755 0.02201612 0.02464101 0.011114520
27    27 0.2331930 0.03403069 0.1142126 0.02196351 0.02482284 0.011192063
28    28 0.2331168 0.03488705 0.1141299 0.02195131 0.02615417 0.011162686
29    29 0.2331186 0.03500649 0.1141960 0.02199822 0.02610196 0.011111177
```

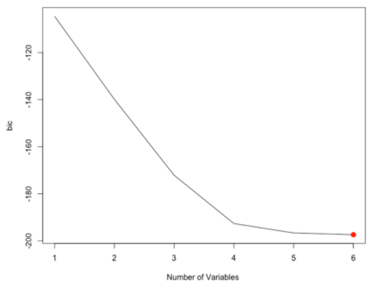
Here as you see RMSE is lower for most of the models. But in model 6 we see higher variance – 0.043 and lower RMSE – 0.23 so it looks the best model for backward subset selection.

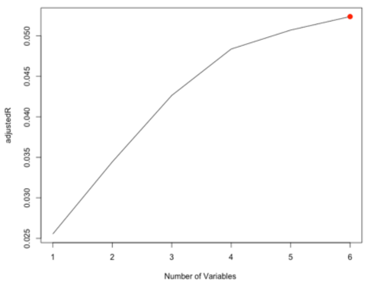


```
         V1  V2  V3  customer_age customer_main_type V6  V7  V8  V9  V10 V11 V12 V13 V14 V15 V16 V17 V18
1  ( 1 ) " " " " " " " "          " "                " " " " " " " " " " " " " " " " " " " " " " " " " "
2  ( 1 ) " " " " " " " "          " "                " " " " " " " " " " " " " " " " " " " " " " " " " "
3  ( 1 ) " " " " " " " "          " "                " " " " " " " " " " " " " " " " " " " " " " " " "*"
4  ( 1 ) " " " " " " " "          " "                " " " " " " " " " " " " " " " " " " " " " " " " "*"
5  ( 1 ) " " " " " " " "          " "                " " " " " " " " " " " " " " " " " " " " " " " " "*"
6  ( 1 ) " " " " " " " "          " "                " " " " " " " " " " " " " " " " " " " " " " " " "*"
         V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38 V39 V40 V41 V42
1  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
2  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
3  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
4  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
5  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
6  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
         V43 V44 V45 V46 V47 V48 V49 V50 V51 V52 V53 V54 V55 V56 V57 V58 V59 V60 V61 V62 V63 V64 V65 V66
1  ( 1 ) " " " " " " " " "*" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
2  ( 1 ) " " " " " " " " "*" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
3  ( 1 ) " " " " " " " " "*" " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
4  ( 1 ) " " " " " " " " "*" " " " " " " " " " " " " " " " " " " " " " " "*" " " " " " " " " " " " " " "
5  ( 1 ) "*" " " " " " " "*" " " " " " " " " " " " " " " " " " " " " " " "*" " " " " " " " " " " " " " "
6  ( 1 ) "*" " " " " "*" "*" " " " " " " " " " " " " " " " " " " " " " " "*" " " " " " " " " " " " " " "
         V67 V68 V69 V70 V71 V72 V73 V74 V75 V76 V77 V78 V79 V80 V81 V82 V83 V84 V85
1  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "
2  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "*" " " " " " "
3  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "*" " " " " " "
4  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "*" " " " " " "
5  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "*" " " " " " "
6  ( 1 ) " " " " " " " " " " " " " " " " " " " " " " " " " " " " " " "*" " " " " " "
> 
```

As we see v18, v43, v46, v47, v59 and v82 is the 6th independent variables model that backward subset choosing. Now would like to see cp, bic and adjustedR-squared to see which model these are choosing.



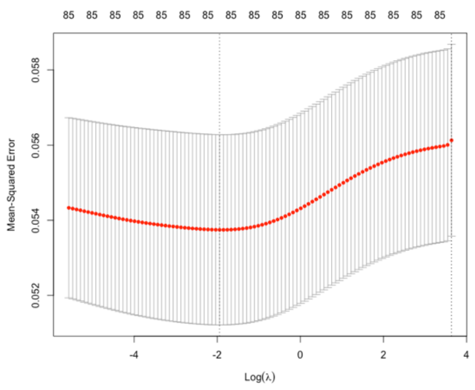




As we see cp, bic and adjusted-r all are doing good on 6 variable model. So, this backward subset is doing best on 6 variable model because it has high variance with low RMSE.

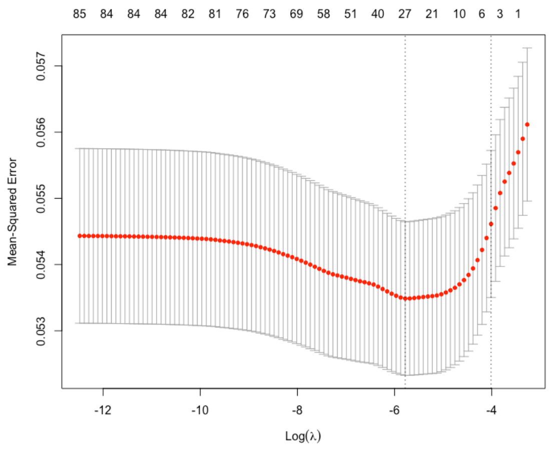
Ridge Regression:

Now we will use another variable selection algorithm name Ridge regression. At first, we run our model on the training data. Here is the plot:



Here we see when log of delta is -2, where MSE error is lowest. This delta penalizing the coefficient and thus way reducing high coefficient variable and thus reducing error. Next we did predict on test data. After prediction on test data we found MSE error for ridge regression. MSE error for ridge regression is 0.0532 which is good. We would like to know if ridge regression could identify categorical response variable correctly but no it just gave all 0 values which is customers who won't buy so it's not doing good on identifying or classifying data correctly.

Lasso Regression:
Next we will use another variable selection algorithm called lasso regression. So first we run the lasso model in our training data.



Here we see when the log of delta is -7, where MSE error is the lowest. Next we did predict to our lasso model to see how it's performing on our dataset. Next we calculated MSE error and we found MSE error for lasso regression is- 0.05342 which is good. Next we saw if lasso predicted classification model correctly but no it just gave all 0's. So, in this model every customer is not buying caravan insurance.

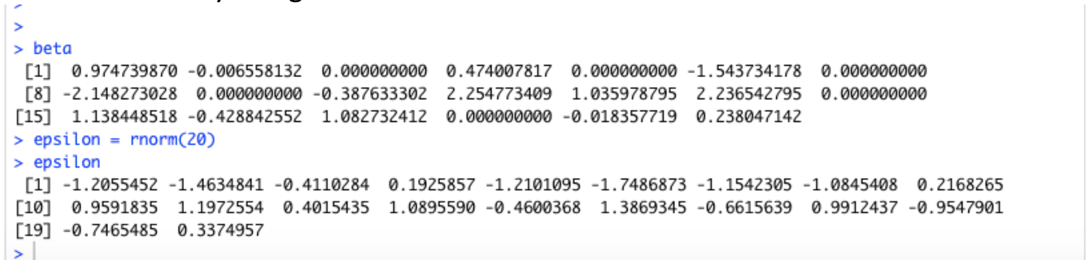
Comparing models:
Ordinary Least Squares: Test RMSE – 0.91, training RMSE – 15.5
Forward Selection: RMSE -0.232

Backward Selection: RMSE – 0.231
Ridge Regression: MSE- 0.0523, RMSE- 0.23
Lasso Regression- MSE-0.0534, RMSE- 0.23

As we can see other variable selection model is doing lot better than Ordinary least squares. Forward, backward, ridge and lasso all selection model is doing good on our data. But as our response variable is categorical these regression models couldn't do so much better as all just predicting customer who didn't buy caravan insurance.

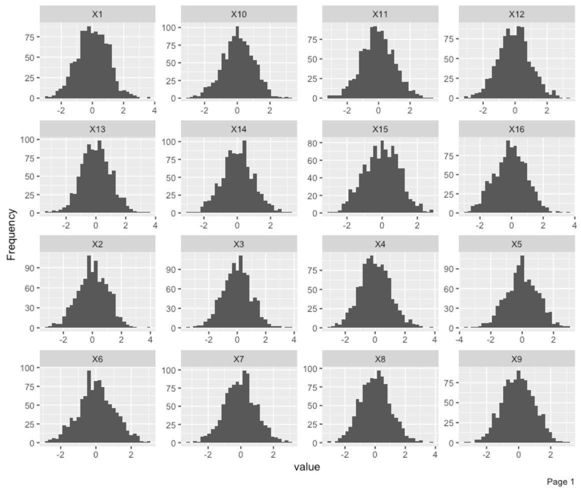
2.

Exploring dataset, preprocessing and Visualization:
Here we generated data set with 20 features and 1 response variable with 1000 rows using rnorm. For response variable we randomly chose our beta value while keeping some them to 0 and for epsilon we also randomly chose our data so we can get the response variable y. After adding response variable, we have 1000 rows with 21 columns. In our data set there is no missing values. Here is the beta values and epsilon chosen randomly using rnorm.



```
>
> beta
 [1]  0.974739870 -0.006558132  0.000000000  0.474007817  0.000000000 -1.543734178  0.000000000
 [8] -2.148273028  0.000000000 -0.387633302  2.254773409  1.035978795  2.236542795  0.000000000
[15]  1.138448518 -0.428842552  1.082732412  0.000000000 -0.018357719  0.238047142
> epsilon = rnorm(20)
> epsilon
 [1] -1.2055452 -1.4634841 -0.4110284  0.1925857 -1.2101095 -1.7486873 -1.1542305 -1.0845408  0.2168265
[10]  0.9591835  1.1972554  0.4015435  1.0895590 -0.4600368  1.3869345 -0.6615639  0.9912437 -0.9547901
[19] -0.7465485  0.3374957
>
```

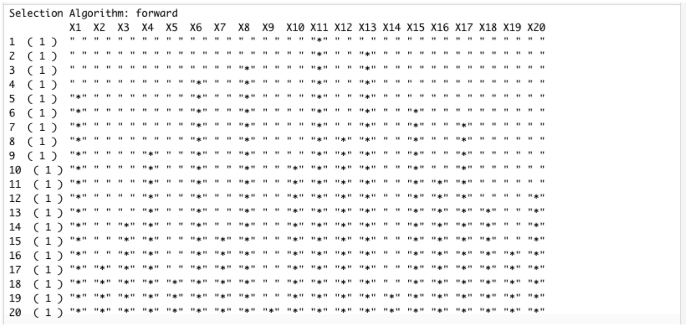
Next we would like to see the histogram to see each variables frequency.



We can see from histogram that most of the features are normally distributed. Next we would like to split our dataset into training and testing. After splitting our dataset, we have 800 training observation and 200 for testing observation.

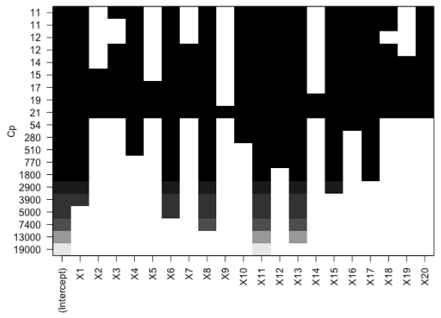
Forward subset selection:

Now we will perform forward subset selection to see which models is the best. Here is the summary after doing forward subset:



```
Selection Algorithm: forward
          X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 X14 X15 X16 X17 X18 X19 X20
1  ( 1 )  " " " " " " " " " " " " " " " " " " " " "*" " " " " " " " " " " " " " " " " " "
2  ( 1 )  " " " " " " " " " " " " " " " " " " " " "*" " " "*" " " " " " " " " " " " " " "
3  ( 1 )  " " " " " " " " " " " " " " "*" " " " " "*" " " "*" " " " " " " " " " " " " " "
4  ( 1 )  " " " " " " " " " " "*" " " "*" " " " " "*" " " "*" " " " " " " " " " " " " " "
5  ( 1 )  "*" " " " " " " " " "*" " " "*" " " " " "*" " " "*" " " " " " " " " " " " " " "
6  ( 1 )  "*" " " " " " " " " "*" " " "*" " " " " "*" " " "*" " " "*" " " " " " " " " " "
7  ( 1 )  "*" " " " " " " " " "*" " " "*" " " " " "*" " " "*" " " "*" " " "*" " " " " " "
8  ( 1 )  "*" " " " " " " " " "*" " " "*" " " " " "*" "*" "*" " " "*" " " "*" " " " " " "
9  ( 1 )  "*" " " " " "*" " " "*" " " "*" " " " " "*" "*" "*" " " "*" " " "*" " " " " " "
10  ( 1 ) "*" " " " " "*" " " "*" " " "*" " " "*" "*" "*" "*" " " "*" " " "*" " " " " " "
11  ( 1 ) "*" " " " " "*" " " "*" " " "*" " " "*" "*" "*" "*" " " "*" "*" "*" " " " " " "
12  ( 1 ) "*" " " " " "*" " " "*" " " "*" " " "*" "*" "*" "*" " " "*" "*" "*" " " " " "*"
13  ( 1 ) "*" " " " " "*" " " "*" " " "*" " " "*" "*" "*" "*" " " "*" "*" "*" "*" " " "*"
14  ( 1 ) "*" " " "*" "*" " " "*" " " "*" " " "*" "*" "*" "*" " " "*" "*" "*" "*" " " "*"
15  ( 1 ) "*" " " "*" "*" " " "*" "*" "*" " " "*" "*" "*" "*" " " "*" "*" "*" "*" " " "*"
16  ( 1 ) "*" " " "*" "*" " " "*" "*" "*" " " "*" "*" "*" "*" " " "*" "*" "*" "*" "*" "*"
17  ( 1 ) "*" "*" "*" "*" " " "*" "*" "*" " " "*" "*" "*" "*" " " "*" "*" "*" "*" "*" "*"
18  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" " " "*" "*" "*" "*" " " "*" "*" "*" "*" "*" "*"
19  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" " " "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
20  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
```

Here we see the best 1 variable model is X11, best 2 variable model is X11 and X13 and then best 3 variable model is X8, X11 and X13. Next we would like to see cp model for our forward subset.

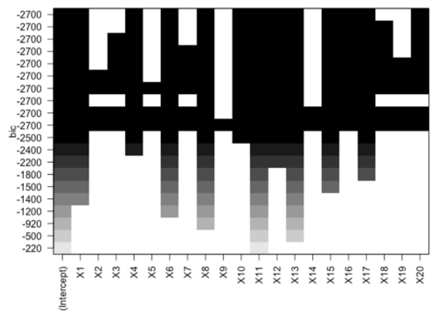


Here we see lowest cp has variable x1, x3, x4, x6, x8, x10, x11, x12, x13, x15, x16, x17, x18 and x20. So, 14 variable model has lowest cp. Here is the coefficient for all this 14 variables model.

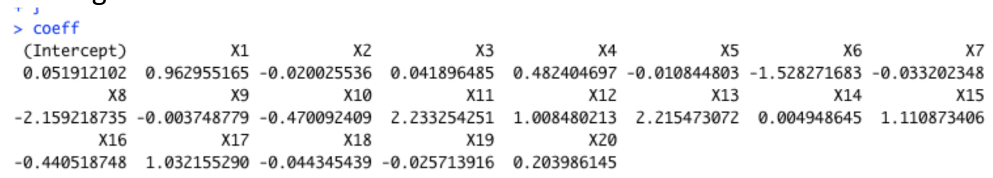
```
> coef(forward_subset ,14)
 (Intercept)          X1          X3          X4          X6          X8         X10         X11 
  0.05378918  0.96298814  0.04172805  0.48340710 -1.52721041 -2.15820579 -0.47088544  2.23147908 
         X12         X13         X15         X16         X17         X18         X20 
  1.01058684  2.21763506  1.11178509 -0.44000420  1.03430914 -0.04620528  0.20315430 
```

As we see for X1 the response variable is increasing by 0.96. Then we see for x6 the response variable is decreasing -1.5. X11 and X13 is highly related with response variable as it's increasing by 2.2 and x8 is most negatively related as response variable is decreasing by -2.15.

Next we would like to see bic to see which model is doing best.

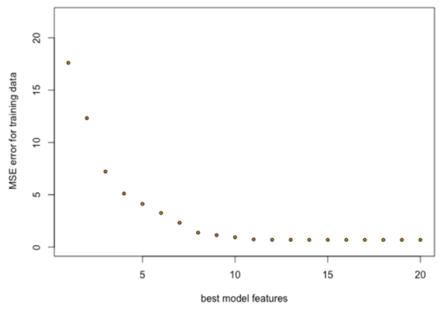


Here we see x1, x4, x6, x8, x10, x11, x12, x13, x15, x16, x17 and x20 has the lowest bic. So, 12 variable model has the lowest bic.
We would like to training coefficient and MSE error for training
Training coefficient:

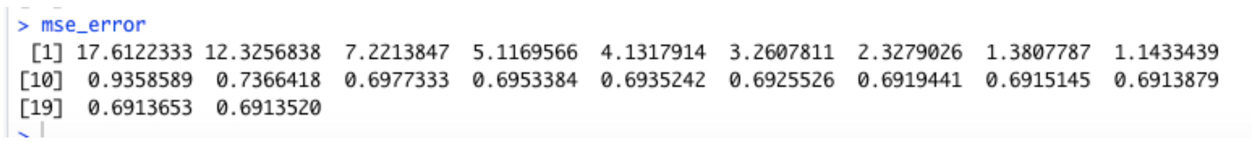


```
> coeff
 (Intercept)           X1           X2           X3           X4           X5           X6           X7
 0.051912102  0.962955165 -0.020025536  0.041896485  0.482404697 -0.010844803 -1.528271683 -0.033202348
          X8           X9          X10          X11          X12          X13          X14          X15
-2.159218735 -0.003748779 -0.470092409  2.233254251  1.008480213  2.215473072  0.004948645  1.110873406
         X16          X17          X18          X19          X20
-0.440518748  1.032155290 -0.044345439 -0.025713916  0.203986145
```

Here we see mostly positively related x11 and x13 with the response variable and most negatively related X8. Next we are going to plot MSE related to each variable model and figure out which model is doing best.



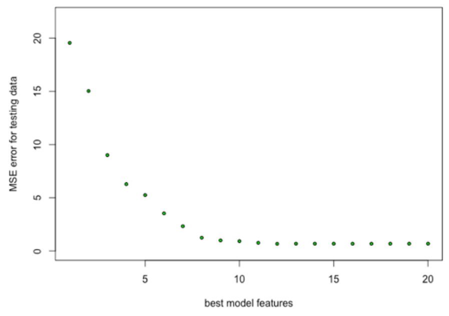
As we see error is getting less from 10 variable models. As variable increases MSE is getting less too. As we can see 1 variable model has the highest MSE. After 11 variable models MSE is decreasing mostly. So, for 20 variable model size the MSE error is the lowest while 1 variable model size MSE error is the highest. Here is the snapshot for the MSE training error list:



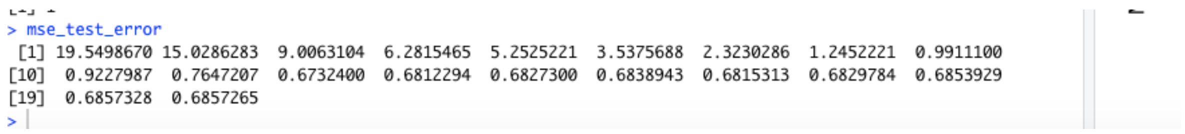
```
> mse_error
 [1] 17.6122333 12.3256838  7.2213847  5.1169566  4.1317914  3.2607811  2.3279026  1.3807787  1.1433439
[10]  0.9358589  0.7366418  0.6977333  0.6953384  0.6935242  0.6925526  0.6919441  0.6915145  0.6913879
[19]  0.6913653  0.6913520
```

Next we would like to see test MSE error:

MSE plot:



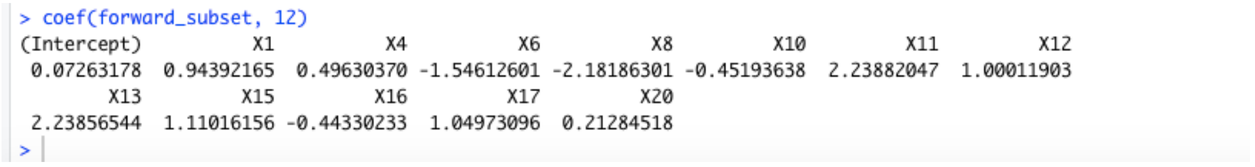
In the test set we also see as size of the variable increasing MSE error is decreasing too. After 10 size model we can see model is doing very good.



```
> mse_test_error
 [1] 19.5498670 15.0286283  9.0063104  6.2815465  5.2525221  3.5375688  2.3230286  1.2452221  0.9911100
[10]  0.9227987  0.7647207  0.6732400  0.6812294  0.6827300  0.6838943  0.6815313  0.6829784  0.6853929
[19]  0.6857328  0.6857265
>
```

As we see here 1 size model has highest MSE error and 12 size model has the lowest MSE error. So, 12 size model is the best for good accuracy in our testing set.

True model coefficient for 12 size model:

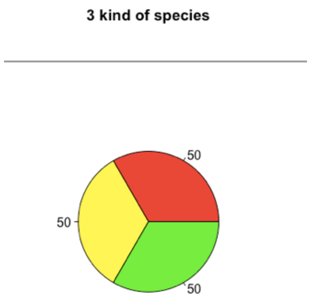


```
> coef(forward_subset, 12)
(Intercept)          X1          X4          X6          X8         X10         X11         X12
 0.07263178  0.94392165  0.49630370 -1.54612601 -2.18186301 -0.45193638  2.23882047  1.00011903
        X13         X15         X16         X17         X20
 2.23856544  1.11016156 -0.44330233  1.04973096  0.21284518
>
```

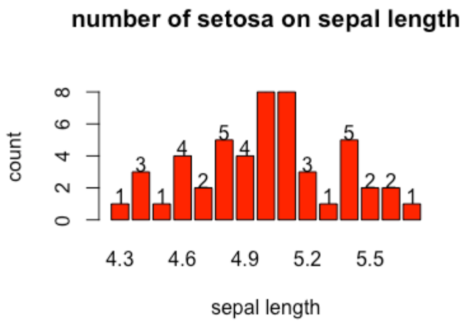
We can see most of the features is related highly either positively or negatively. So, thus this model will give us good accuracy while lowering MSE. As we can see response variable is getting increased 0.94 by feature X1. We also see y variable is decreasing -2.18 by feature X8 and so on.

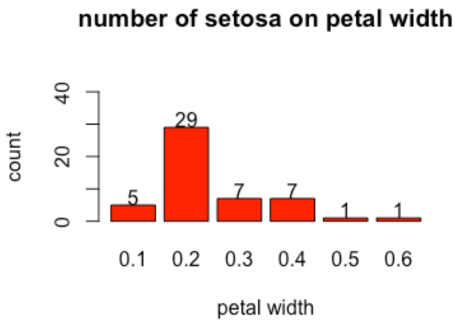
3. Exploring dataset and Visualization:

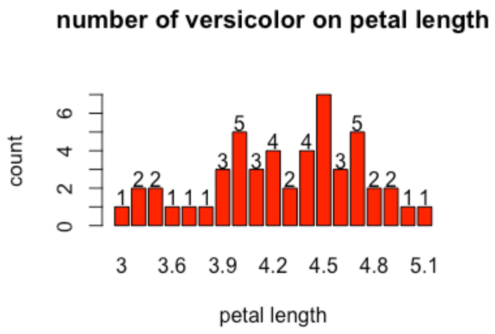
In the Iris dataset there is 150 rows and 5 columns. Dependent variable is categorical called ‘Species’. There is 3 kind of category called ‘setosa’, ‘versicolor’ and ‘virginica’ in the dependent categorical variable. In this dataset there is no missing values. In the dependent variable there is 50 setosa, 50 vericular and 50 virginica. Here is the piechart:



Next we would like to see type of flower species and see numbers based on sepal length, width and petal length, width.

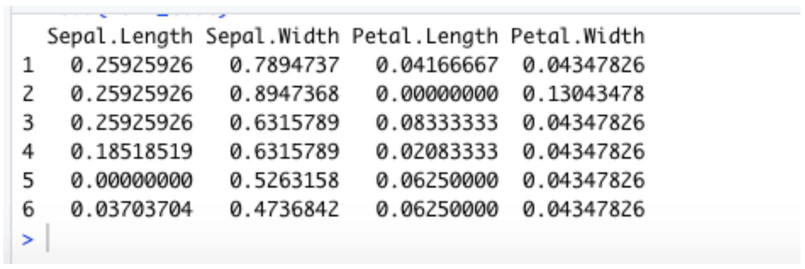






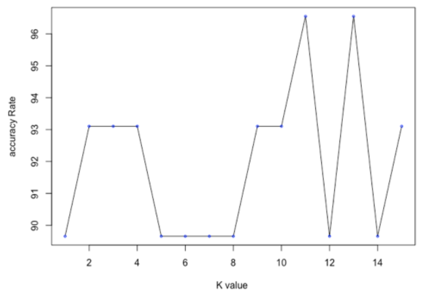
Here in 2nd plot we see petal width for 0.2cm there is 29 setosa, we see on the 3rd plot there is like 5 versicolor for 3.9-4.0cm petal length.

Data cleaning and transformation:
First, we would like to split our dataset into training and testing. After splitting 80% for training and 20% for testing we got 121 rows with 5 columns for training and 29 rows with 5 columns for testing. Next we would like to normalize our data so the ranges get between 0 and 1. Here the data looks like now:

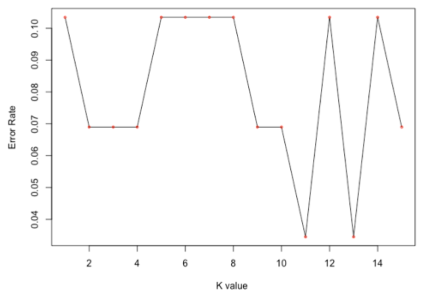


```
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1   0.25925926   0.7894737   0.04166667  0.04347826
2   0.25925926   0.8947368   0.00000000  0.13043478
3   0.25925926   0.6315789   0.08333333  0.04347826
4   0.18518519   0.6315789   0.02083333  0.04347826
5   0.00000000   0.5263158   0.06250000  0.04347826
6   0.03703704   0.4736842   0.06250000  0.04347826
>
```

3(a). For KNN we are using k values 1 through 15. We are fitting KNN on training data and then predicting on the test data. After prediction we would like to see accuracy and error rate.

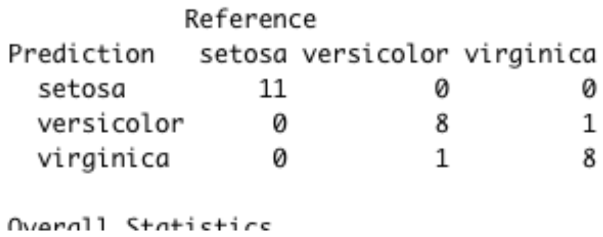


As we see accuracy gets higher when k value gets higher. We see here k = 11 has the highest accuracy rate but then again at k = 12 accuracy gets low and at this point overfitting occurs.



As we see When k values are low the error rate is higher. As we can see when k = 1 the error rate is high. Because of small k value it can't capture all data so it can't classify categorical data correctly so error rate gets high. Here k = 11 is the best because it has lowest error. But then we can see after k = 11 the plot is going up and down and it's because of overfitting happening here.

Next we will see the confusion matrix to see the species:

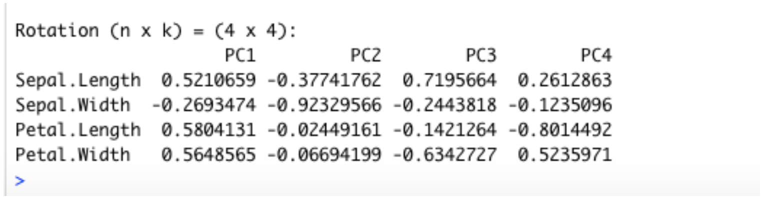

```
             Reference
Prediction   setosa versicolor virginica
  setosa         11          0         0
  versicolor      0          8         1
  virginica       0          1         8

Overall Statistics
```

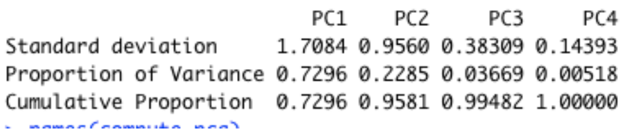
This confusion matrix is from test data set. So here 29 species. Let's see if it correctly predicted data from this confusion matrix. So, for true setosa there is 11 and it predicted 11. For versicolor there is 8 versicolor and it predicted 8 too. There is 1

virginica but it predicted as versicolor. There is 8 true virginica and predicted 8. There is 1 true versicolor but predicted as virginica.
So total data = 29 and correctly predicted 27. So KNN did good on classifying data correctly.

3(b). First, we applied principle component analysis on the iris data set. After applying it looks like:

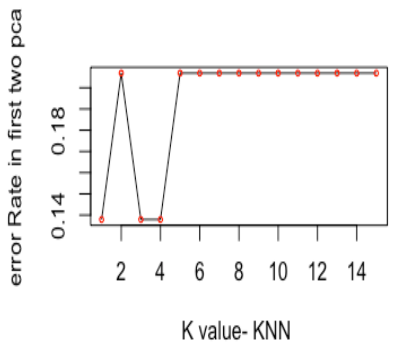


```
Rotation (n x k) = (4 x 4):
                    PC1         PC2        PC3        PC4
Sepal.Length  0.5210659 -0.37741762  0.7195664  0.2612863
Sepal.Width  -0.2693474 -0.92329566 -0.2443818 -0.1235096
Petal.Length  0.5804131 -0.02449161 -0.1421264 -0.8014492
Petal.Width   0.5648565 -0.06694199 -0.6342727  0.5235971
>
```

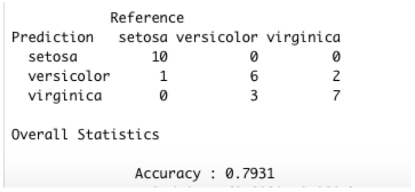


```
                          PC1    PC2     PC3     PC4
Standard deviation     1.7084 0.9560 0.38309 0.14393
Proportion of Variance 0.7296 0.2285 0.03669 0.00518
Cumulative Proportion  0.7296 0.9581 0.99482 1.00000
```

Here we see for PC2 the proportion of variance is 0.2285 which is better than PC3 and PC4. Now we would like to perform KNN for first two principle component analysis dataset. So now we would like to split our dataset where training is 80% and testing is 20%. Next we ran our KNN on PC2. We found this error plot:



We see here when k value is 1,3 and 4 error rate is lowest. After k value = 5 the error rate increased and its same till 15th k-value.
Next I would like to see confusion matrix to get the idea about if this model classified data correctly.



```
            Reference
Prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      1          6         2
  virginica       0          3         7

Overall Statistics

               Accuracy : 0.7931
```

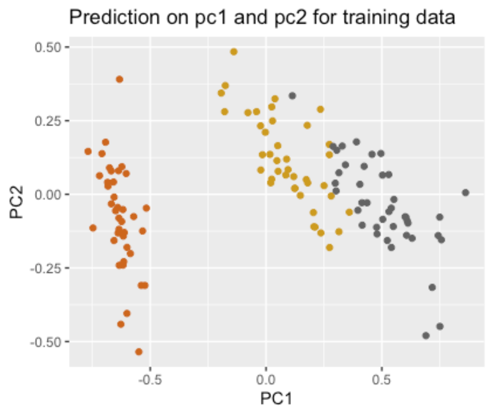
Here we see accuracy is 79%. Then we see from confusion matrix model predicted 1 data as versicolor but the real is setosa. There is 3 virginica predicted but real value would versicolor. Total data here is 29. Correctly predicted 23 and misclassified 6 data.

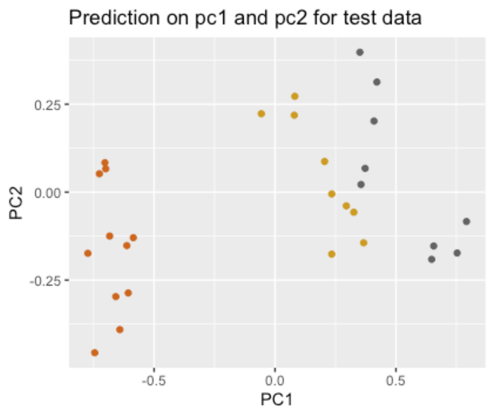
Comparing error rate from part(a):

In part 'a' we saw than when K values get higher error gets lower and then it overfitted. When we performed KNN on two principle components the error gets lower when k values are low. And error rate got higher when K values gets higher. And we see in part(a) correctly predicted data was 27 out of 29 where in two principle KNN we see correctly predicted data was 23. So, part(a) KNN doing good in classifying. PCA would be good for large dataset and as this dataset aren't large so it's not giving better accuracy than part(a).

Plotting First two PCA:

Next we will plot scores for first two PCA and then we will color the samples by class.





We can see with this 2 pca we see the separation of three species for both training and testing data. In the training data there is 80% variance in PC1 and 12% variance in PC2. This two pca clearly classified data into 3 species.