Lab6diagnosis

```
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
v ggplot2 3.3.6      v purrr   0.3.4
v tibble  3.1.8      v dplyr   1.0.10
v tidyr   1.2.1      v stringr 1.4.1
v readr   2.1.2      v forcats 0.5.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
Loading required package: lattice

Attaching package: 'caret'

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

    lift
```

```
# load data, do some pre-processing
cells <- read_csv('https://raw.githubusercontent.com/idc9/course-materials/main/3-predicti
```

```
Rows: 569 Columns: 31
-- Column specification --------------------------------------------------------
Delimiter: ","
```

```
chr  (1): diagnosis
dbl (30): radius_mean, texture_mean, perimeter_mean, area_mean, smoothness_m...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# make diagnosis a factor -- this is required to make the code below work!
cells <- cells %>%
    mutate(diagnosis=as.factor(diagnosis))

cells
```

```
# A tibble: 569 x 31
   diagnosis radius_mean textu~1 perim~2 area_~3 smoot~4 compa~5 conca~6 conca~7
   <fct>           <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1 M                18.0    10.4   123.    1001   0.118   0.278   0.300   0.147
 2 M                20.6    17.8   133.    1326   0.0847  0.0786  0.0869  0.0702
 3 M                19.7    21.2   130     1203   0.110   0.160   0.197   0.128
 4 M                11.4    20.4    77.6    386.  0.142   0.284   0.241   0.105
 5 M                20.3    14.3   135.    1297   0.100   0.133   0.198   0.104
 6 M                12.4    15.7    82.6    477.  0.128   0.17    0.158   0.0809
 7 M                18.2    20.0   120.    1040   0.0946  0.109   0.113   0.074
 8 M                13.7    20.8    90.2    578.  0.119   0.164   0.0937  0.0598
 9 M                13      21.8    87.5    520.  0.127   0.193   0.186   0.0935
10 M                12.5    24.0    84.0    476.  0.119   0.240   0.227   0.0854
# ... with 559 more rows, 22 more variables: symmetry_mean <dbl>,
#   fractal_dimension_mean <dbl>, radius_se <dbl>, texture_se <dbl>,
#   perimeter_se <dbl>, area_se <dbl>, smoothness_se <dbl>,
#   compactness_se <dbl>, concavity_se <dbl>, concave.points_se <dbl>,
#   symmetry_se <dbl>, fractal_dimension_se <dbl>, radius_worst <dbl>,
#   texture_worst <dbl>, perimeter_worst <dbl>, area_worst <dbl>,
#   smoothness_worst <dbl>, compactness_worst <dbl>, concavity_worst <dbl>, ...
```

Part 1: Understanding and Exploring the Data

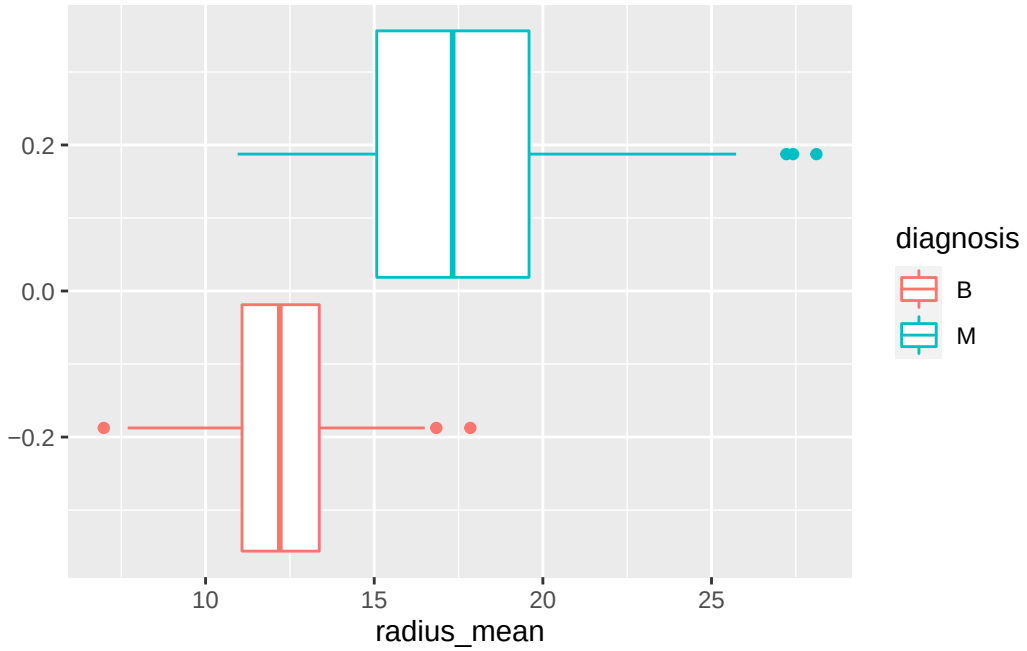
1. **How many classes are in diagnosis?**

There are 2 classes in this diagnosis, either malignant or benign

2. **Use a box plot to compare the `radius_mean` for benign vs. malignant biopsies. What is the takeaway from this plot?**

The main takeaway from this plot is that the Malignant cases have a radius mean that's way bigger than that of the benign cases. We see how the max values for the radius means is way bigger for malignant cases than for the benign cases. We also see that malignant cases have a bit more variability when it comes to radius mean, while benign cases have radius means that are withing a smaller window.

```
cells%>%
  ggplot(aes(x=radius_mean,
             color=diagnosis))+
geom_boxplot()
```

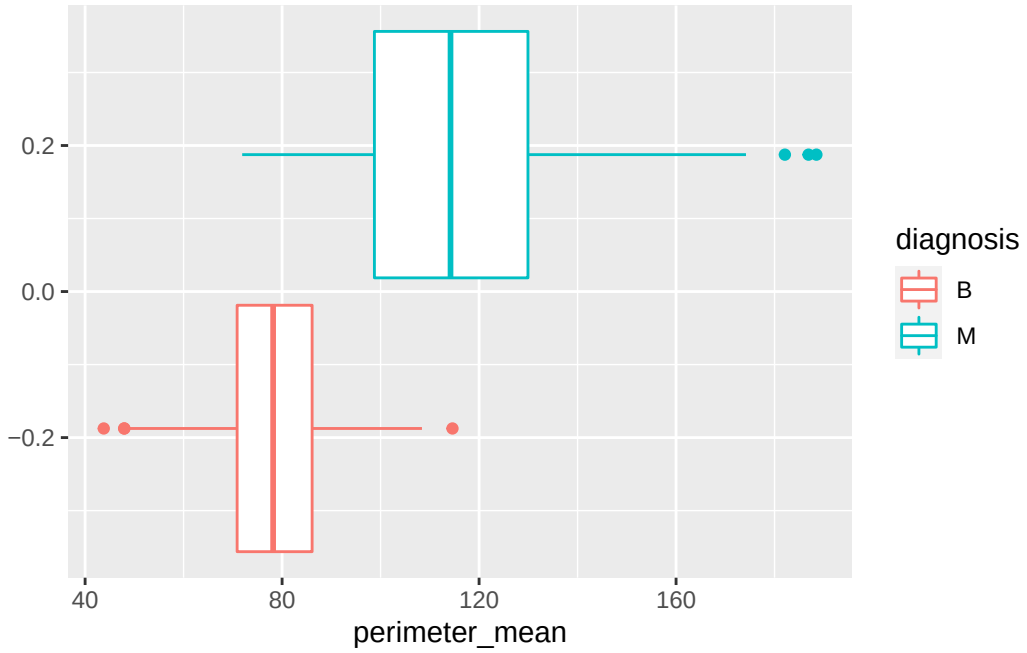


3. **Repeat the previous question for another variable of your choosing. What is the interpretation?**

In here I am calculating the relationship between perimeter mean and the values in malignant and benign cases. Its clear how the perimeter of the malignant cells have a bigger perimeters than those benign cells.

The main takeaway from this plot is that the Malignant cases have a perimeter mean that's way bigger than that of the benign cases. We see how the max values for the perimeter means is way bigger for malignant cases than for the benign cases. We also see that malignant cases have a bit more variability when it comes to perimeter mean, while benign cases have perimeter means that are withing a smaller window.

```
cells%>%
  ggplot(aes(x=perimeter_mean,
             color=diagnosis))+
geom_boxplot()
```

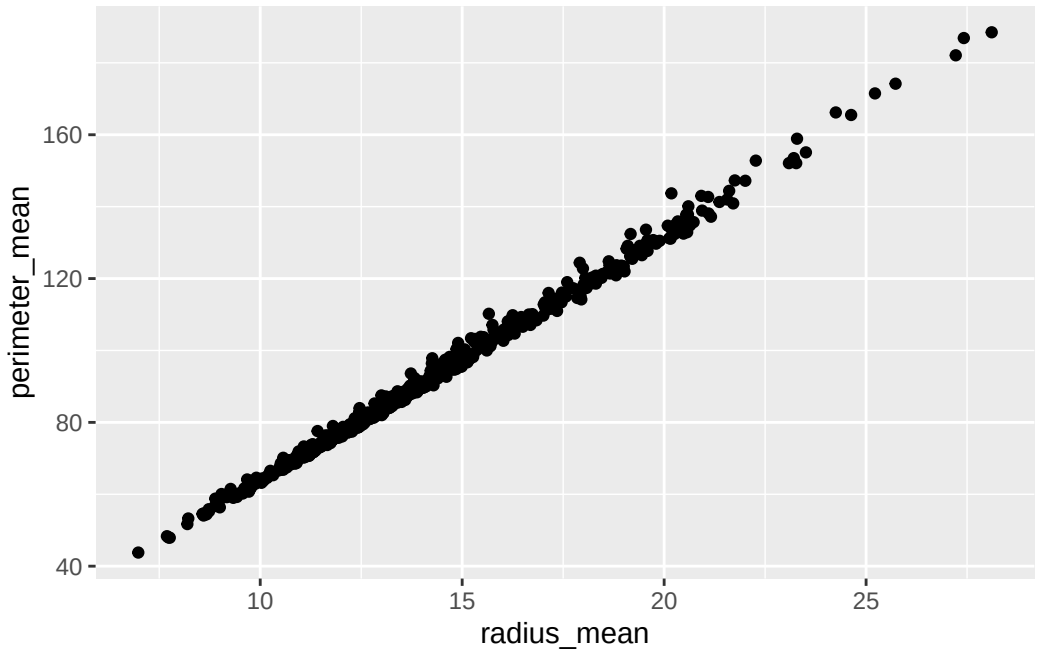


4. **Make a plot that examines the association between `radius_mean` and `perimeter_mean`. Calculate the correlation between these two variables. Is there a strong association between these two variables? Does this make sense?**

The correlation between these two variables is 0.9978553.

This is a very strong correlation between these two variables, which makes sense, because perimeter and radius are proportionally related, so as one increases so does the other. This also matches the graph, as the linear relation is clearly very strong and upwards sloping.

```
cells%>%
  ggplot(aes(x=radius_mean,
             y=perimeter_mean)) +
  geom_point()
```




```
cells%>%
summarise(cor(x=radius_mean,
              y=perimeter_mean,
              use="everything"))
```

```
# A tibble: 1 x 1
  `cor(x = radius_mean, y = perimeter_mean, use = "everything")`
                                                           <dbl>
1                                                          0.998
```

5. **Make a single plot that examines the association between `radius_mean` and `radius_se` separately for each diagnosis. Hint: `aes()` should have three arguments. Calculate the correlation between these two variables for each diagnosis.**

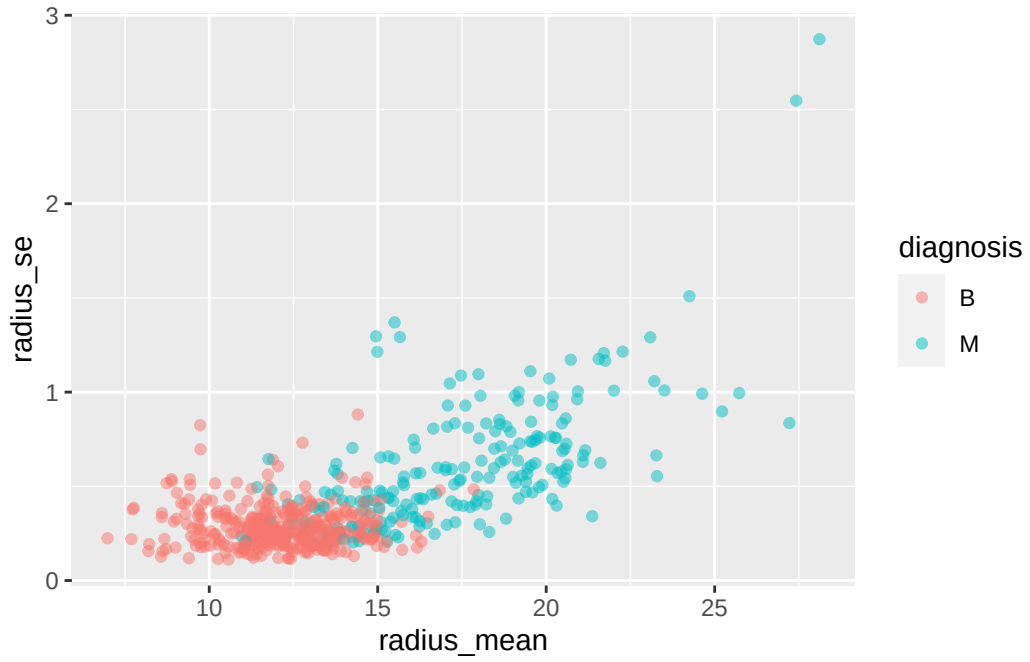
The results show that the relation between these two variables is very different depending on whether the case is benign or malignant. With benign cases, there is no strong correlation between these two variables, as we see the number is -0.027761.

For the correlation between these two variables in malignant cases, there is a stronger correlations than there is for benign cases. Here the correlation is moderate, at 0.639269

The way I could explain this would be that with some benign cancer cases, not all cells grow at the same rate nor following a size average, which is why there is no clear correlation

between mean and sd. With malignant cases however, the cancer spreads very quickly and the cells become big with time, which is why these two increase together and are moderatly correlated.

```
cells %>%
      ggplot(aes(x=radius_mean,
                 y=radius_se,
                 color=diagnosis)) +
      geom_point(alpha=0.5)
```



Give an interpretation of these results. In particular comment on: Is the relationship between `radius_mean` and `radius_se` different for benign biopsies vs. malignant biopsies? If so, can you give an explanation for this difference?

```
cells %>%
group_by(diagnosis) %>%
  summarize(correlation_b=cor(radius_mean,
                              radius_se,
                              use="everything"))
```

```
# A tibble: 2 x 2
  diagnosis correlation_b
```

```
  <fct>             <dbl>
1 B               -0.0278
2 M                0.639
```

Part II: Classification with K-nearest-neighbors

6. **Split the full cells data set into an 80/20 train/test set split. Name the training data frame `train_df` and the test data frame `test_df`.**

```
set.seed(1234)
train_indices <- createDataPartition(y = cells$diagnosis,
                                     p=0.8,
                                     list=FALSE)


train_df <- cells %>%
  slice(train_indices)
test_df <-  cells %>%
  slice(-train_indices)
```

7. **Fit a KNN model with K=1 to the training data. Evaluate the training accuracy and test accuracy of this model. Hint: using the `knn3` function from the `caret`package as in the notes.**

As seen in the models, the accuracy for train data with K=1 equals to 1 & the accuracy for the test data with K=1 equals 0.9026549

```
#TRAIN
knn_1 <- knn3(diagnosis ~ ., k = 1, data = train_df)
     diagnosis_pred_train = predict(knn_1,
                                    newdata = train_df,
                                    type = 'class')
train_df %>%
mutate(diagnosis_pred = diagnosis_pred_train,
       correct_prediction = diagnosis_pred == diagnosis) %>%
summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1        1
```

```
#TEST
knn_1 <- knn3(diagnosis ~ ., k = 1, data = train_df)
     diagnosis_pred_test = predict(knn_1,
                                   newdata = test_df,
                                   type = 'class')
test_df %>%
  mutate(diagnosis_pred = diagnosis_pred_test,
         correct_prediction = diagnosis_pred == diagnosis) %>%
  summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.903
```

8. **Repeat the previous question with K=20.**

As seen in the models, the accuracy for train data with K=20 equals to 0.932017 & the accuracy for the test data with K=20 equals 0.9115044

```
#TRAIN
knn_20 <- knn3(diagnosis ~ ., k = 20, data = train_df)
     diagnosis_pred_train = predict(knn_20,
                                   newdata = train_df,
                                   type = 'class')
train_df %>%
mutate(diagnosis_pred = diagnosis_pred_train,
       correct_prediction = diagnosis_pred == diagnosis) %>%
summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.932
```

```
#TEST
knn_1 <- knn3(diagnosis ~ ., k = 20, data = train_df)
     diagnosis_pred_test = predict(knn_1,
                                   newdata = test_df,
                                   type = 'class')
```

```r
test_df %>%
  mutate(diagnosis_pred = diagnosis_pred_test,
         correct_prediction = diagnosis_pred == diagnosis) %>%
  summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.912
```

9. **Standardize the training and test data with mean centering and standard deviation scaling. Make sure column the mean/standard deviations are obtained from `train_df`. Create new data frames called `train_stand` and `test_stand`. Hint: use `preProcess()` from the `caret` package as in the notes.**

```r
standardize_params <- preProcess(train_df,
                                 method = c("center", "scale"))
```

```r
train_stand <- predict(standardize_params, train_df)
train_stand
```

```
# A tibble: 456 x 31
   diagnosis radius_mean textu~1 perim~2 area_~3 smoot~4 compa~5 conca~6 conca~7
   <fct>           <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1 M               1.05   -2.05    1.22    0.928  1.54    3.25    2.62     2.50
 2 M               1.76   -0.354   1.62    1.82  -0.796  -0.474  -0.0251   0.541
 3 M               1.52    0.444   1.51    1.48   0.932   1.05    1.34     2.01
 4 M              -0.763   0.245  -0.590  -0.756  3.22    3.37    1.89     1.43
 5 M              -0.478  -0.828  -0.390  -0.507  2.20    1.24    0.853    0.814
 6 M               1.12    0.153   1.09    1.04  -0.109   0.0941  0.294    0.639
 7 M              -0.476   1.08   -0.334  -0.510  1.56    2.54    1.71     0.930
 8 M               0.507   0.900   0.415   0.372 -0.982  -0.697  -0.693   -0.400
 9 M               1.38    1.26    1.60    1.26   0.0839  2.65    1.46     1.60
10 M               0.460   1.06    0.455   0.330 -0.847  -0.0705  0.129    0.120
# ... with 446 more rows, 22 more variables: symmetry_mean <dbl>,
#   fractal_dimension_mean <dbl>, radius_se <dbl>, texture_se <dbl>,
#   perimeter_se <dbl>, area_se <dbl>, smoothness_se <dbl>,
#   compactness_se <dbl>, concavity_se <dbl>, concave.points_se <dbl>,
#   symmetry_se <dbl>, fractal_dimension_se <dbl>, radius_worst <dbl>,
#   texture_worst <dbl>, perimeter_worst <dbl>, area_worst <dbl>,
```

```
#   smoothness_worst <dbl>, compactness_worst <dbl>, concavity_worst <dbl>, ...
```

```
test_stand <- predict(standardize_params, test_df)
test_stand
```

```
# A tibble: 113 x 31
   diagnosis radius_mean textu~1 perim~2 area_~3 smoot~4 compa~5 conca~6 conca~7
   <fct>           <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
 1 M               1.69   -1.14   1.71     1.74   0.285    0.539  1.35    1.41
 2 M              -0.130   0.348 -0.0849  -0.231  1.58     1.13   0.0586  0.278
 3 M              -0.326   0.575 -0.193   -0.390  2.16     1.67   1.20    1.14
 4 M               0.441  -0.326  0.451    0.326  0.0631   0.472  0.131   0.436
 5 B              -0.177  -1.14  -0.195   -0.263  0.111   -0.424 -0.276  -0.0284
 6 B              -1.29   -1.58  -1.28    -1.06   0.431   -0.730 -0.735  -0.717
 7 M               0.308   1.37   0.403    0.192  0.835    1.23   0.983   0.983
 8 M               1.23    1.33   1.30     1.17   0.709    1.58   1.77    1.92
 9 M              -0.194   0.345 -0.157   -0.282  0.376    0.403  0.215   0.139
10 M              -0.892   0.467 -0.817   -0.797  1.84     0.334  0.192   0.198
# ... with 103 more rows, 22 more variables: symmetry_mean <dbl>,
#   fractal_dimension_mean <dbl>, radius_se <dbl>, texture_se <dbl>,
#   perimeter_se <dbl>, area_se <dbl>, smoothness_se <dbl>,
#   compactness_se <dbl>, concavity_se <dbl>, concave.points_se <dbl>,
#   symmetry_se <dbl>, fractal_dimension_se <dbl>, radius_worst <dbl>,
#   texture_worst <dbl>, perimeter_worst <dbl>, area_worst <dbl>,
#   smoothness_worst <dbl>, compactness_worst <dbl>, concavity_worst <dbl>, ...
```

10. **Verify the columns of `train_stand` have means of 0 and standard deviations of 1. What about `test_stand`?**

We can see in the models bellow that the means for the train stand are all very close to 0, practically 0. The negative e exponent just means there are a lot of zeros in front of the number, but its basically 0.

For the standard deviation of the train stand, we see it is indeed 1 in all collumns.

```
#TRAIN STAND
train_stand %>%
          select(-diagnosis) %>%
          summarize_all(mean)
```

```
# A tibble: 1 x 30
  radius_mean texture_m~1 perime~2 area_m~3 smooth~4 compac~5 concav~6 concave~7
        <dbl>       <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>     <dbl>
1    1.61e-16    2.28e-16 2.63e-16 5.91e-18 3.84e-16 8.43e-17 8.06e-17 -7.13e-17
# ... with 22 more variables: symmetry_mean <dbl>,
#   fractal_dimension_mean <dbl>, radius_se <dbl>, texture_se <dbl>,
#   perimeter_se <dbl>, area_se <dbl>, smoothness_se <dbl>,
#   compactness_se <dbl>, concavity_se <dbl>, concave.points_se <dbl>,
#   symmetry_se <dbl>, fractal_dimension_se <dbl>, radius_worst <dbl>,
#   texture_worst <dbl>, perimeter_worst <dbl>, area_worst <dbl>,
#   smoothness_worst <dbl>, compactness_worst <dbl>, concavity_worst <dbl>, ...
```

```
#TRAIN STAND
train_stand%>%
     select(-diagnosis) %>%
     summarize_all(sd)
```

```
# A tibble: 1 x 30
  radius_mean texture_~1 perim~2 area_~3 smoot~4 compa~5 conca~6 conca~7 symme~8
        <dbl>      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1           1          1       1       1       1       1       1       1       1
# ... with 21 more variables: fractal_dimension_mean <dbl>, radius_se <dbl>,
#   texture_se <dbl>, perimeter_se <dbl>, area_se <dbl>, smoothness_se <dbl>,
#   compactness_se <dbl>, concavity_se <dbl>, concave.points_se <dbl>,
#   symmetry_se <dbl>, fractal_dimension_se <dbl>, radius_worst <dbl>,
#   texture_worst <dbl>, perimeter_worst <dbl>, area_worst <dbl>,
#   smoothness_worst <dbl>, compactness_worst <dbl>, concavity_worst <dbl>,
#   concave.points_worst <dbl>, symmetry_worst <dbl>, ...
```

We can see in the models bellow that the means for the test stand are close to 0, but never 0, in fact these are farther apart from 0 than those values on the train stand. In here the values range from -0.0768927 to 0.05867621 and many more.

For the standard deviation of the test stand, we see it is very close but never exactly 1, the values range from 0.8556344 to 0.883732 and many others. These values are farther apart from 1 than those values found in the train stand.

```
#TEST STAND
test_stand %>%
     select(-diagnosis) %>%
     summarize_all(mean)
```

```
# A tibble: 1 x 30
  radius_mean textur~1 perim~2 area_~3 smoot~4 compa~5 concav~6 concav~7 symme~8
        <dbl>    <dbl>   <dbl>   <dbl>   <dbl>   <dbl>    <dbl>    <dbl>   <dbl>
1     -0.0769  -0.0271 -0.0710 -0.0999  0.0587  0.0350 -0.00797 -7.48e-4 -0.0717
# ... with 21 more variables: fractal_dimension_mean <dbl>, radius_se <dbl>,
#   texture_se <dbl>, perimeter_se <dbl>, area_se <dbl>, smoothness_se <dbl>,
#   compactness_se <dbl>, concavity_se <dbl>, concave.points_se <dbl>,
#   symmetry_se <dbl>, fractal_dimension_se <dbl>, radius_worst <dbl>,
#   texture_worst <dbl>, perimeter_worst <dbl>, area_worst <dbl>,
#   smoothness_worst <dbl>, compactness_worst <dbl>, concavity_worst <dbl>,
#   concave.points_worst <dbl>, symmetry_worst <dbl>, ...
```

```
#TEST STAND
test_stand %>%
        select(-diagnosis) %>%
        summarize_all(sd)
```

```
# A tibble: 1 x 30
  radius_mean texture_~1 perim~2 area_~3 smoot~4 compa~5 conca~6 conca~7 symme~8
        <dbl>      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1       0.856      0.931   0.854   0.802   0.884   0.941   0.938   0.944    1.05
# ... with 21 more variables: fractal_dimension_mean <dbl>, radius_se <dbl>,
#   texture_se <dbl>, perimeter_se <dbl>, area_se <dbl>, smoothness_se <dbl>,
#   compactness_se <dbl>, concavity_se <dbl>, concave.points_se <dbl>,
#   symmetry_se <dbl>, fractal_dimension_se <dbl>, radius_worst <dbl>,
#   texture_worst <dbl>, perimeter_worst <dbl>, area_worst <dbl>,
#   smoothness_worst <dbl>, compactness_worst <dbl>, concavity_worst <dbl>,
#   concave.points_worst <dbl>, symmetry_worst <dbl>, ...
```

11. **Compute the KNN train and test set accuracy with K=1 and K=20 for the standardized data.**

As we see in the models, when K=1, in the standardized data for train, accuracy is 1 & and in the test data accuracy is 0.9292035

As we see in the models, when K=20, in the standardize data for train, accuracy is 0.9649123 & and in the test data accuracy is 0.9469027.

```
#K=1 TRAIN
knn_1 <- knn3(diagnosis ~ ., k = 1, data = train_stand)
  diagnosis_pred_train = predict(knn_1,
                                 newdata = train_stand,
```

```
                                    type = 'class')
  train_stand %>%
      mutate(diagnosis_pred = diagnosis_pred_train,
      correct_prediction = diagnosis_pred == diagnosis) %>%
      summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1        1
```

```
  #K=1 TEST
  knn_1 <- knn3(diagnosis ~ ., k = 1, data = train_stand)
    diagnosis_pred_test = predict(knn_1,
                                  newdata = test_stand,
                                  type = 'class')
  test_stand %>%
    mutate(diagnosis_pred = diagnosis_pred_test,
    correct_prediction = diagnosis_pred == diagnosis) %>%
    summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.929
```

```
  #K=20 TRAIN
  knn_20 <- knn3(diagnosis ~ ., k = 20, data = train_stand)
     diagnosis_pred_train = predict(knn_20,
                                  newdata = train_stand,
                                  type = 'class')
  train_stand %>%
    mutate(diagnosis_pred = diagnosis_pred_train,
          correct_prediction = diagnosis_pred == diagnosis) %>%
    summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
```

```
    <dbl>
1    0.969
```

```
#K=20 TEST
knn_20 <- knn3(diagnosis ~ ., k = 20, data = train_stand)
  diagnosis_pred_test = predict(knn_20,
                                newdata = test_stand,
                                type = 'class')
test_stand %>%
mutate(diagnosis_pred = diagnosis_pred_test,
         correct_prediction = diagnosis_pred == diagnosis) %>%
summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.938
```

12. **Using cross-validation with 5 folds, pick the best value of K for KNN. Evaluate every value of K between 1 and 30. Plot the curve showing validation accuracy vs. K. For the selected value of K, what is the training and test set accuracy?**

As seen in the model and after seeing k=13 is the best value (according to accuracy)

The accuracy of these train and test stands with K=13 are the following:

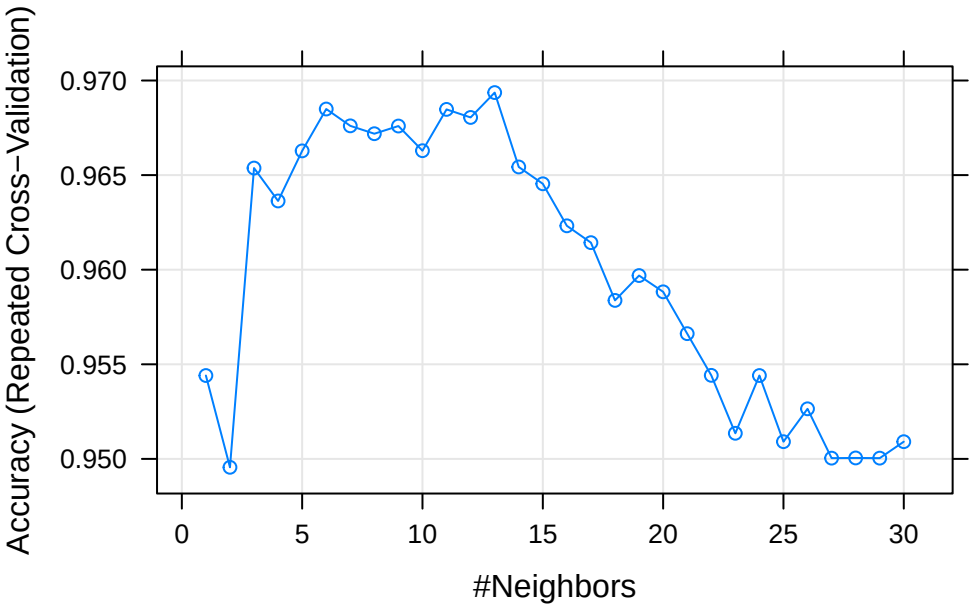
Train stand=0.9714912

Test stand=0.9646018

```
set.seed(393)

k_values_to_try <- seq(from = 1, to = 30)
training_control <- trainControl(method = "repeatedcv", repeats = 5)
knn_cv <- train(diagnosis~ . ,
                data = train_stand,
                method = "knn",
                trControl = training_control,
                metric = "Accuracy",
                tuneGrid = data.frame(k = k_values_to_try))

plot(knn_cv)
```




```
knn_cv
```

```
k-Nearest Neighbors

456 samples
 30 predictor
  2 classes: 'B', 'M'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 410, 410, 411, 411, 410, 411, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   1  0.9544058  0.9023044
   2  0.9495556  0.8918874
   3  0.9653720  0.9250320
   4  0.9636329  0.9210488
   5  0.9662802  0.9265983
   6  0.9684928  0.9313251
   7  0.9676039  0.9293646
   8  0.9671884  0.9284634
   9  0.9675942  0.9294001
```

```
  10  0.9662899  0.9264028
  11  0.9684734  0.9311400
  12  0.9680483  0.9301151
  13  0.9693623  0.9328465
  14  0.9654300  0.9241520
  15  0.9645411  0.9222627
  16  0.9623188  0.9173648
  17  0.9614300  0.9153823
  18  0.9583768  0.9084954
  19  0.9596908  0.9114365
  20  0.9588309  0.9094391
  21  0.9566184  0.9044854
  22  0.9544155  0.8996733
  23  0.9513527  0.8925127
  24  0.9544058  0.8994509
  25  0.9509082  0.8915307
  26  0.9526473  0.8954066
  27  0.9500386  0.8895367
  28  0.9500483  0.8896446
  29  0.9500386  0.8895388
  30  0.9509082  0.8914786

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 13.
```

```r
#K=13 TRAIN
knn_13 <- knn3(diagnosis ~ ., k = 13, data = train_stand)
  diagnosis_pred_test = predict(knn_13,
                                newdata = train_stand,
                                type = 'class')
train_stand %>%
  mutate(diagnosis_pred = diagnosis_pred_test,
         correct_prediction = diagnosis_pred == diagnosis) %>%
  summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.971
```

```r
#K=13 TEST
knn_13 <- knn3(diagnosis ~ ., k = 13, data = train_stand)
  diagnosis_pred_test = predict(knn_13,
                                newdata = test_stand,
                                type = 'class')
test_stand %>%
   mutate(diagnosis_pred = diagnosis_pred_test,
          correct_prediction = diagnosis_pred == diagnosis) %>%
   summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.965
```

13. **Pick any 3 variables and build a classification model that you train with cross-validation based on only these three variables from `train_stand`. What is the test set accuracy?**

In this question, I chose the following 3 variables: radius_mean, smoothness_mean and texture_mean

In this case, the accuracy is 0.9026549. In this stand the test error is 0.0973451

```r
knn_variables <- knn3(diagnosis ~ radius_mean +
                                  smoothness_mean +
                                  texture_mean,
                      k = 13,
                      data = train_stand)
diagnosis_pred_test = predict(knn_variables,
                              newdata = test_stand,
                              type = 'class')
test_stand %>%
  mutate(diagnosis_pred = diagnosis_pred_test,
         correct_prediction = diagnosis_pred == diagnosis) %>%
summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.903
```

```
1-0.9026549
```

```
[1] 0.0973451
```