Lab6

Vivian Yeh

Part 1: Understanding and Exploring the Data

```
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
```

```
library(ggplot2)
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
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>, ...
```

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

   There are 2 classes in diagnosis.

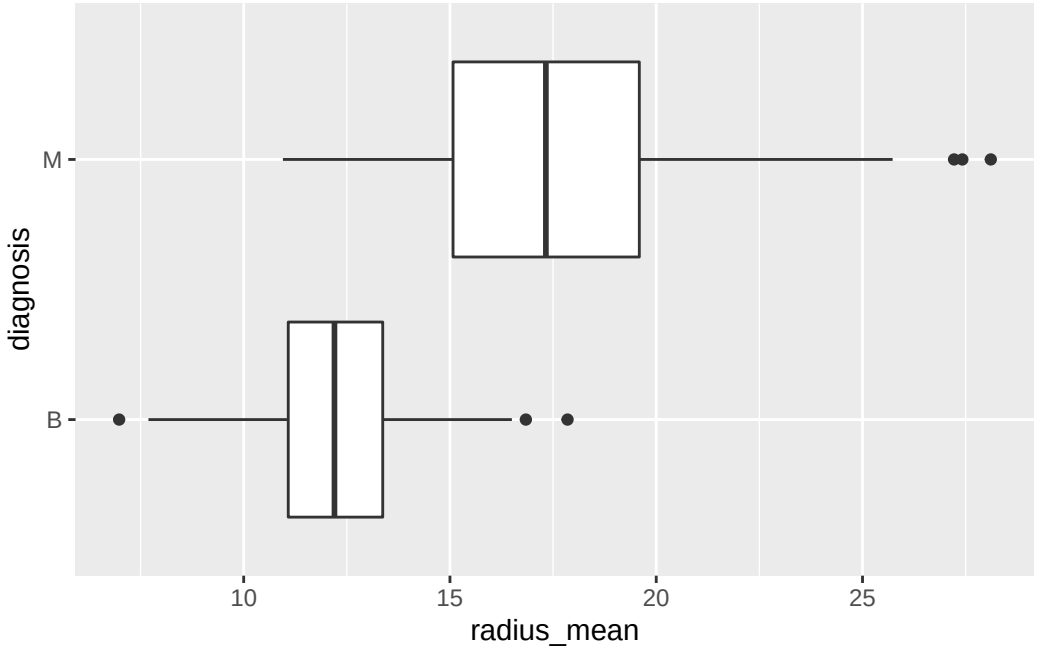
```r
cells %>%
  count(diagnosis)
```

```
# A tibble: 2 x 2
  diagnosis     n
  <fct>     <int>
1 B           357
2 M           212
```

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

   Most of the members of the group with malignant biopsies have a higher radius mean of the biopsies, the radius mean is higher than the group of benign biopsies.

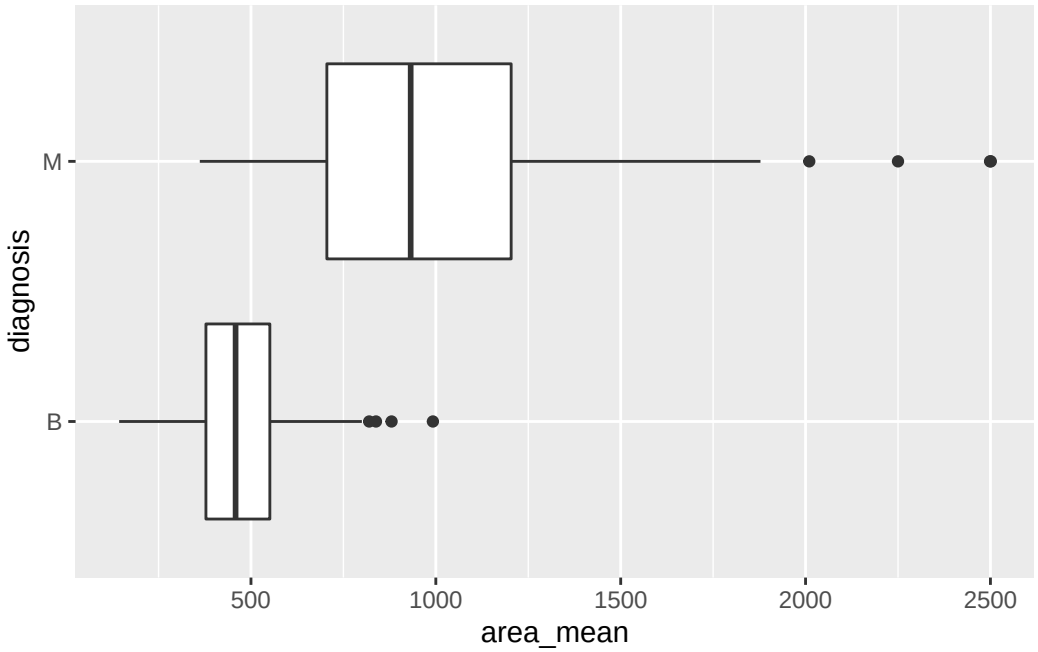
```
cells %>%
  ggplot(aes(x = radius_mean, y = diagnosis)) +
  geom_boxplot()
```



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

   Most of the members of the group with malignant biopsies, the area mean is higher than the group of benign biopsies.

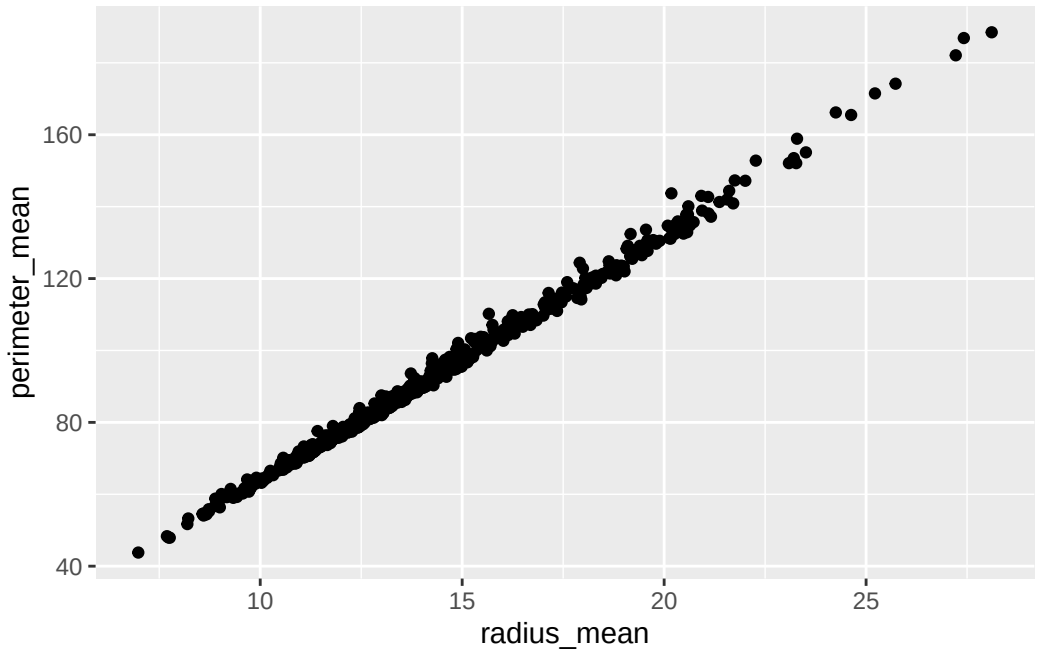
```
cells %>%
  ggplot(aes(x = area_mean, y = diagnosis)) +
  geom_boxplot()
```



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 radius_mean and perimeter_mean is 0.9978553, and this indicates the same interpretation as the plot, which is there is a strong association between these two variables. This does make sense because the higher correlation, the stronger association will be shown on the plot.

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

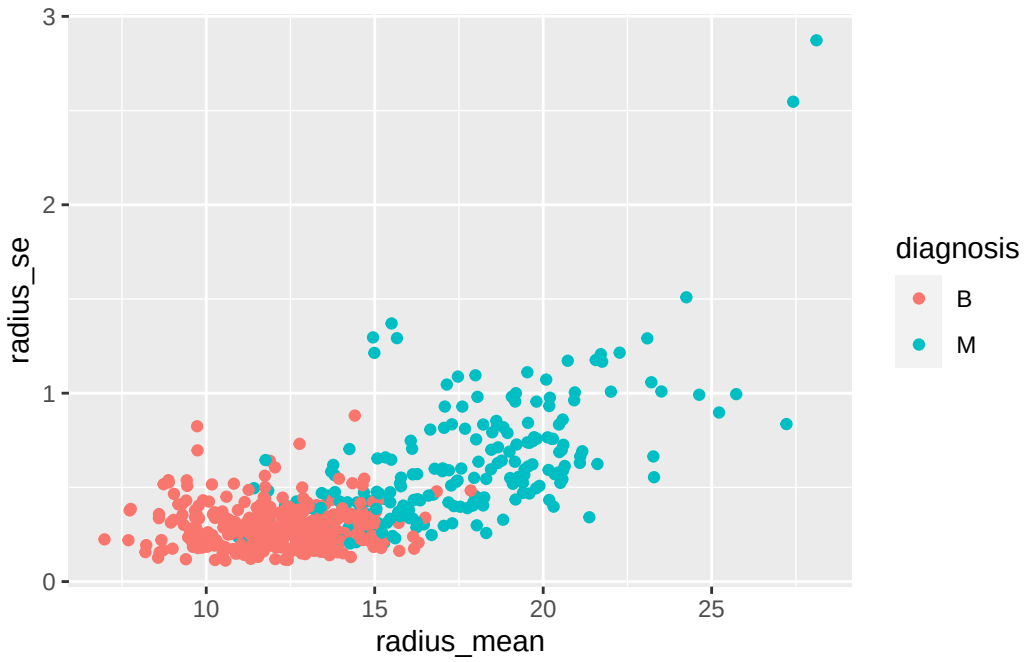



```
cells %>%
  summarize(cor1 = cor(radius_mean, perimeter_mean))
```

```
# A tibble: 1 x 1
   cor1
  <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.**

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




```
  cells %>%
    filter(diagnosis == 'M') %>%
    summarize(cor_M = cor(radius_mean, perimeter_mean))
```

```
# A tibble: 1 x 1
  cor_M
  <dbl>
1 0.995
```

```
  cells %>%
    filter(diagnosis == 'B') %>%
    summarize(cor_B = cor(radius_mean, perimeter_mean))
```

```
# A tibble: 1 x 1
  cor_B
  <dbl>
1 0.997
```

The relationship between radius_mean and radius_se different for benign biopsies is more like no association because no matter the radius_mean low or high, the radius_se keeps the same range of the data, which has a density from 0 to 0.5. However, they have a 0.9952815 correlation coefficient.

The relationship between radius_mean and radius_se different for malignant biopsies is a weak positive association because when the radius_mean increases, the radius_se also grows up. Moreover, their correlation coefficient is 0.9967688, which is higher than the benign biopsies group.

To explain this difference, the plot represents that for benign biopsies the radius_se and radius_mean, the radius_mean does not affect the radius_se data.

Part 2: 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`.

   Make sure you use the seed in the code chunk shown below.

```
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.

   The accuracy for training data is 1, and the accuracy for test data is 0.9026549.

```
knn_fit_1 <- knn3(diagnosis ~ ., k = 1, data = train_df)

# training accuracy
dia_pred_train = predict(knn_fit_1,
                         newdata = train_df,
                         type = 'class')
train_df %>%
    mutate(dia_pred = dia_pred_train,
           correct_prediction = dia_pred == diagnosis) %>%
```

```
    summarize(accuracy = mean(correct_prediction))
```

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

```
  # test accuracy
  dia_pred_test = predict(knn_fit_1,
                          newdata = test_df,
                          type = 'class')
  test_df %>%
      mutate(dia_pred = dia_pred_test,
             correct_prediction = dia_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.

   The accuracy for training data is 0.9320175, and the accuracy for test data is 0.9115044.

```
  knn_fit_20 <- knn3(diagnosis ~ ., k = 20, data = train_df)

  # training accuracy
  dia_pred_train_8 = predict(knn_fit_20,
                          newdata = train_df,
                          type = 'class')
  train_df %>%
      mutate(dia_pred = dia_pred_train_8,
             correct_prediction = dia_pred == diagnosis) %>%
      summarize(accuracy = mean(correct_prediction))
```

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

```
# test accuracy
dia_pred_test_8 = predict(knn_fit_20,
                          newdata = test_df,
                          type = 'class')
test_df %>%
    mutate(dia_pred = dia_pred_test_8,
           correct_prediction = dia_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.

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

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

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

    The columns of train_stand have means of 0 and standard deviations of 1.

    The columns of test_stand do not have means of 0 and standard deviations of 1.

```
# training set
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 %>%
    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>, ...
```

```
# test set
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 %>%
    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.

    For K = 1, the accuracy for training data is 1, and the accuracy for test data is 0.9292035.

    For K = 20, the accuracy for training data is 1, and the accuracy for test data is 0.9292035

```
# K = 1
# Training set accuracy
dia_pred_train_11 = predict(knn_fit_1,
                            newdata = train_stand,
                            type = 'class')

train_stand %>%
    mutate(dia_pred = dia_pred_train_11,
           correct_prediction = dia_pred == diagnosis) %>%
    summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.627
```

```
# Test set accuracy
dia_pred_test_11 = predict(knn_fit_1,
                           newdata = test_stand,
```

```
                              type = 'class')

  test_stand %>%
      mutate(dia_pred_test = dia_pred_test_11,
             correct_prediction = dia_pred_test == diagnosis) %>%
      summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.628
```

```
  # K = 20
  # Training set accuracy
  dia_pred_train_11_20 = predict(knn_fit_20,
                              newdata = train_stand,
                              type = 'class')

  train_stand %>%
      mutate(dia_pred = dia_pred_train_11_20,
             correct_prediction = dia_pred == diagnosis) %>%
      summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.627
```

```
  # Test set accuracy
  dia_pred_test_11_20 = predict(knn_fit_20,
                              newdata = test_stand,
                              type = 'class')

  test_stand %>%
      mutate(dia_pred = dia_pred_test_11_20,
             correct_prediction = dia_pred == diagnosis) %>%
      summarize(accuracy = mean(correct_prediction))
```

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

```
1     0.628
```

12. Using cross-validation, 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?

    This shows that the most accurate predictions came from using K = 13. For K = 13, the training set accuracy is 0.9693623, and the test set accuracy is 0.9646018.

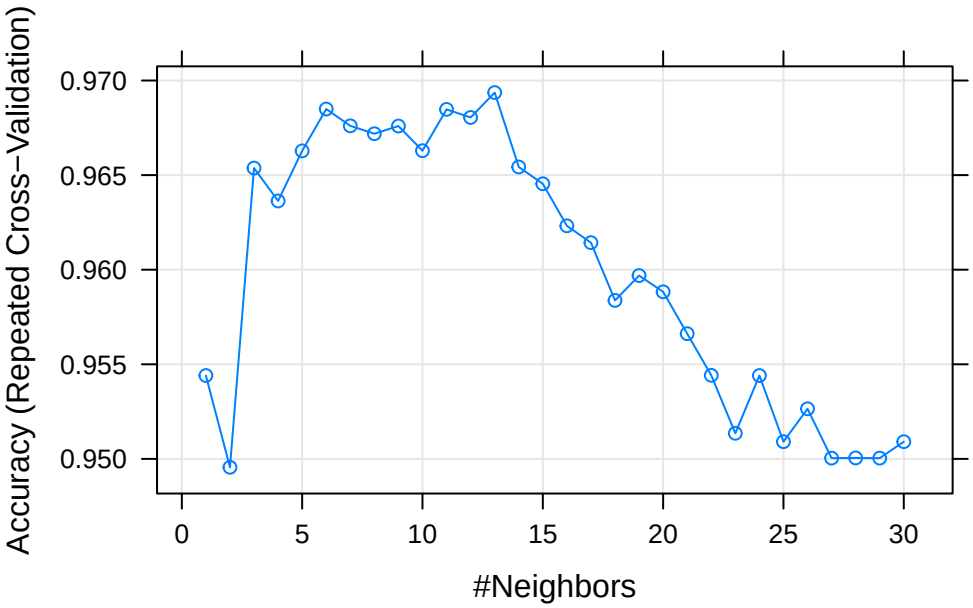
```r
set.seed(393)

# create a numeric vector with values from 1 to 30
k_values_to_try <- seq(from = 1, to = 30)

# setup the tuning parameter selection procedure
training_control <- trainControl(method = "repeatedcv", repeats = 5)

# try all values of k then fit the best to the full training data set
knn_cv <- train(diagnosis ~ .,
                data = train_stand,
                method = "knn",
                trControl = training_control,
                metric = "Accuracy",
                tuneGrid = data.frame(k = k_values_to_try))

# Plot the curve showing validation accuracy vs K
plot(knn_cv)
```




```
# training set accuracy
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.
```

```
# get test predictions
dia_pred_test_cv = predict(knn_cv, newdata = test_stand)

# compute the accuracy for test set
test_stand %>%
    mutate(dia_pred = dia_pred_test_cv) %>%
    mutate(correct_prediction = dia_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 based on only these three variables from `train_stand`. What is the test set error?

I picked concavity_mean, area_mean, and texture_mean to build a classification model from train_stand. The test set error is 0.8849558.

```
knn_fit_13 <- knn3(diagnosis ~ concavity_mean + area_mean + texture_mean, k = 1, data

# process the test data
test_stand <- predict(standardize_params, test_df)

# get test predictions
dia_pred_test_13 = predict(knn_fit_13,
                           newdata = test_stand,
                           type = 'class')

# compute test set accuracy
test_stand %>%
    mutate(dia_pred = dia_pred_test_13,
           correct_prediction = dia_pred == diagnosis) %>%
    summarize(accuracy = mean(correct_prediction))
```

```
# A tibble: 1 x 1
  accuracy
     <dbl>
1    0.885
```