project 1: Classification model that Predict Iris Species

shayma

2024-10-28

IMPORTING LIBRARIES AND DATA SET - For this project, we will be analyzing the Iris data set, specifically creating an ml model that predicts which Species a given flower will be based on other variables in the data set. Specifically, we will analyze how variables, like Petal length and Sepal length can be used to determine which species a flower will be and how much those variables affects the model's predictability.

To begin, we will import the necessary libraries that contain the data set and the ml functions. dplyr includes the 'iris' data set that we will be using. Additionally, I have imported ggplot2 to create interactive plots when doing an initial analysis of the data. caret is a necessary package for classification ml models, which we will be conducting today.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```
data("iris")
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

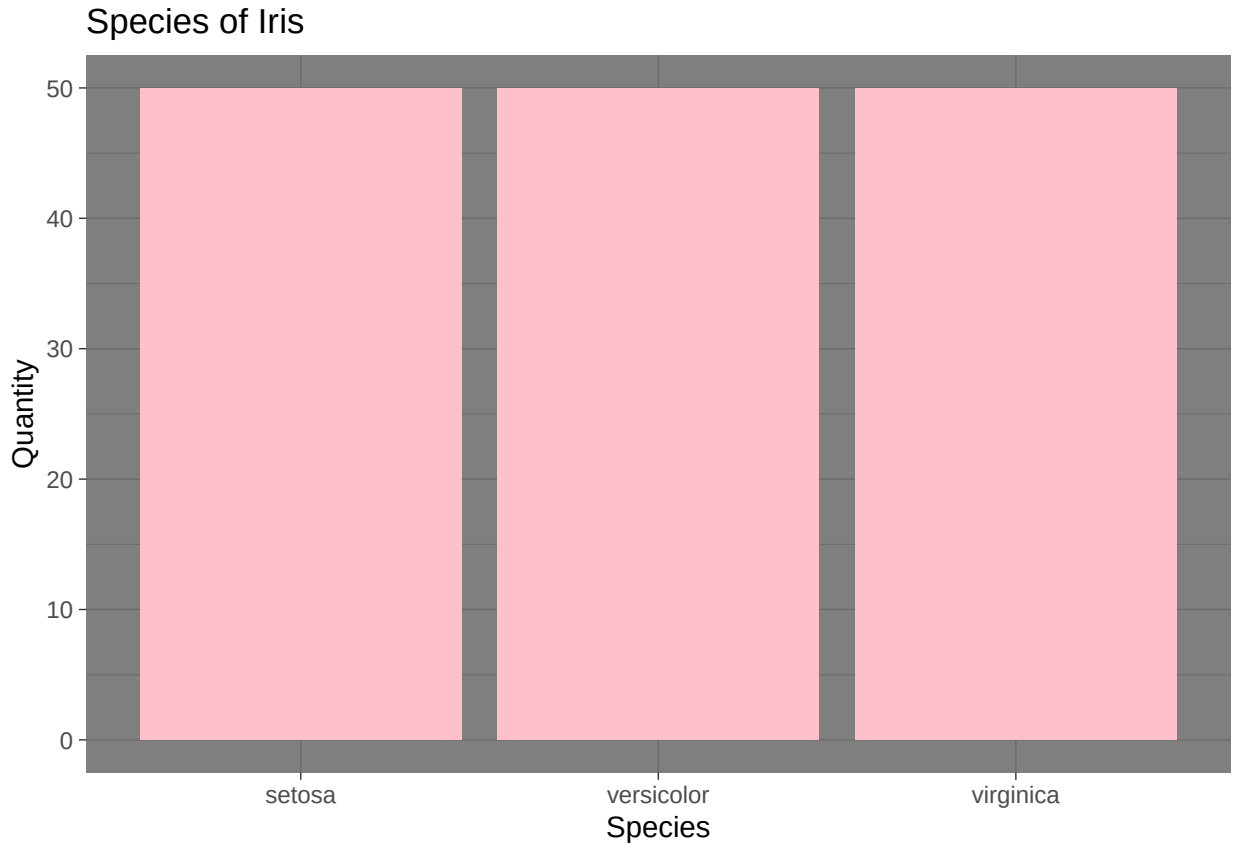
OVERVIEW OF 'IRIS' - 'iris' is a data set that includes various variables regarding three different iris species. These variables include Sepal Length, Sepal width, and Petal Lengths and widths. This data is generally straightforward for the questions we will be asking the ml model to predict. By this I mean that there is no missing data to filter out when we start our analysis and we have both categorical and numeric variablesto make predictions.

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

DATA EXPLORATION- Since we want to predict which factors contribute most to species categorization, we will start by plotting some variables included in the data set and how those compare across the three species. To start, we have simply graphed the quantity of each species available in the data frame. This is important, as it shows us if the model will have enough information to calculate one species over another. For example, if the data isn't normally distributed, there may be some errors that occur when we actually predict which species a flower will be.

```
ggplot(data = iris, aes(x = Species)) +
         geom_bar(fill = "pink") +
  labs(title = "Species of Iris",
       x = "Species",
       y = "Quantity") + theme_dark()
```



PLOTTING SPECIFIC VARIABLES- Next, we have plotted the variable "Sepal.Length" to start gathering an idea of which variables we think will most contribute to differentiation between species. In this case, we can clearly see that the sepal length seems to be very different across the species, with each type of iris having a very different median sepal length and little over lapping data. To start, it seems that that this variable may have a big impact on how the species are categorized when we create a prediction classification model.

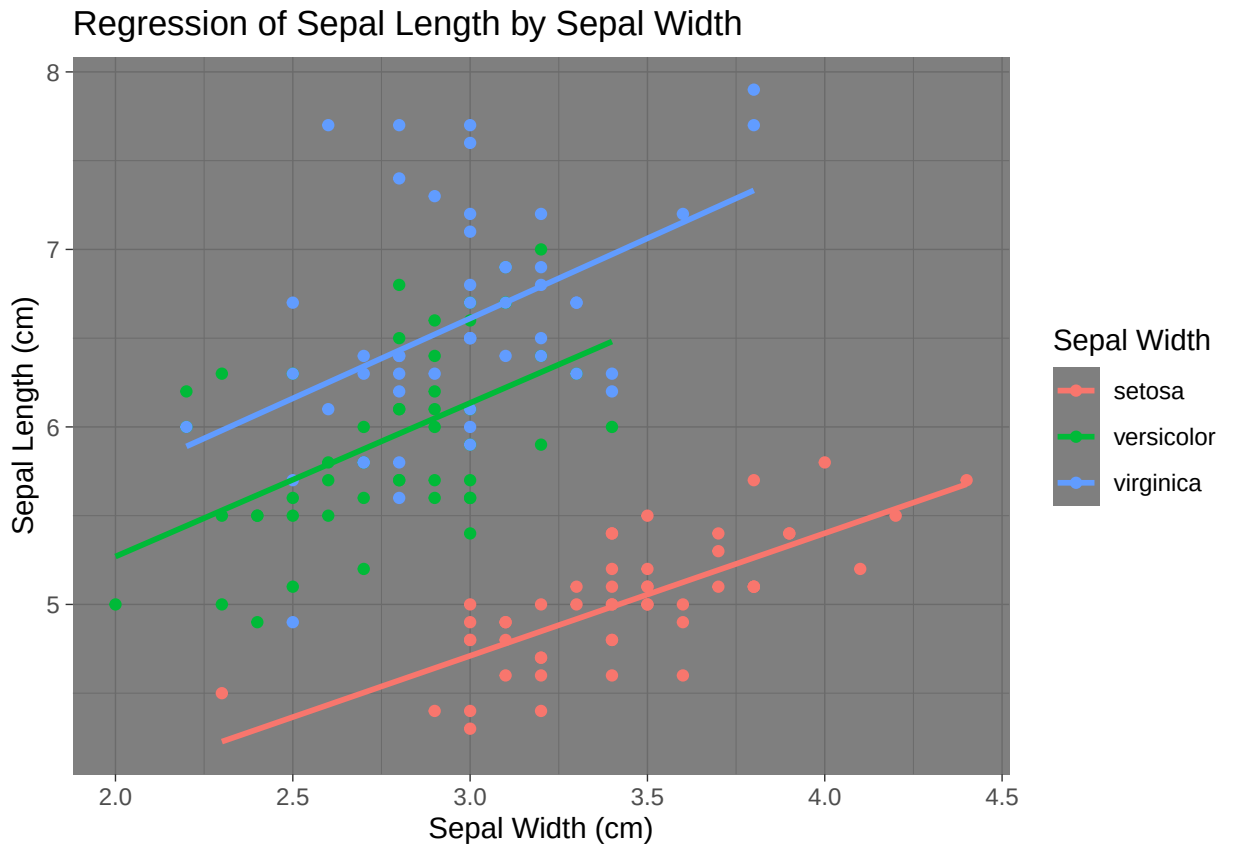
```
ggplot(iris, aes(x = Species, y = Sepal.Length, color = Species)) +
  geom_boxplot(fill = 'pink') +
  labs(title = "Comparison of Sepal Length by Species",
       x = "Species",
       y = "Sepal Length") +
  theme_dark()
```

REGRESSION- To further explore the data, we have created a regression plot detailing the sepal lengths and widths for each flower in a given species. As we can determine from the regression below, we see that there is a positive relationship between sepal length and width for each of the species. Additionally, it is clear that some of the data seems stronger in the Setosa species. While it has a a regression with a smaller slope, meaning that the strength of a positive relationship between Sepal Length and Width is smaller than the other two species, we can clearly see that the data doesn't over lap with that of Versicolor and Virginica. As I will explian when we plot the Classification model's accuracy, the overlapping and close-knit data between the Versicolor and Viriginica species may result in less accurate predictions by the model.

```
ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length, color = Species)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(title = "Regression of Sepal Length by Sepal Width",
       x = "Sepal Width (cm)",
```

```
       y = "Sepal Length (cm)",
       color = "Sepal Width") +
  theme_dark()
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



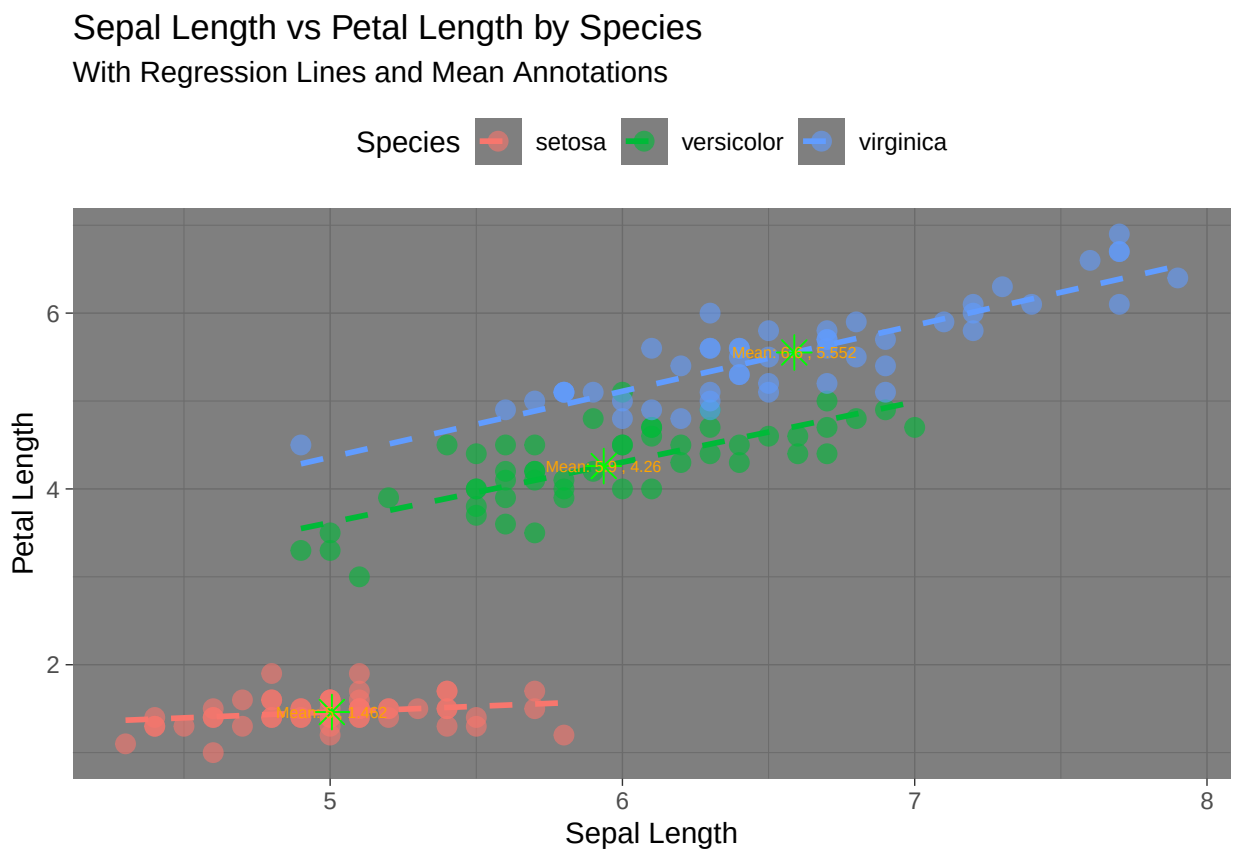
DATA RANGLING- This Plot displays the aggregated differences between Sepal length and petal length for each of the three species, as well as the mean difference for each species. For this specific project, the differences between the aggregated means and the regression model can tell us just how varied the data is. By this, I mean that variances further from the mean may be considered outliers and cause confusion for the classification model's ability to predict which species a specific flower is. For example, we can see that for the species Virginica (outlined in blue) one flower has a petal length of ~4.5 and sepal length of ~4.75. However, another flower of the same species has a petal length of ~6.5 and sepal length of ~7.75. We must be cognizant of these trends going into the model prediction phase, as it will help explain the strength of model's accuracy. Since the model uses the sepal length and petal length in it's prediction algorithim, these variances can cause issues in it's accuracy to predict the correct species.

```
iris_means = iris %>%
  group_by(Species) %>%
  summarise(
    Mean_Sepal.Length = mean(Sepal.Length),
    Mean_Sepal.Width = mean(Sepal.Width),
    Mean_Petal.Length = mean(Petal.Length),
    Mean_Petal.Width = mean(Petal.Width)
  )
```

```r
ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, color = Species)) +
  geom_point(alpha = 0.6, size = 3) +
  geom_smooth(method = "lm", se = FALSE, linetype = "dashed", size = 1) +
  geom_point(data = iris_means, aes(x = Mean_Sepal.Length, y = Mean_Petal.Length),
             shape = 8, size = 4, color = "green") +
  geom_text(data = iris_means, aes(x = Mean_Sepal.Length, y = Mean_Petal.Length,
                                   label = paste("Mean:", round(Mean_Sepal.Length, 1), ",", round(Mean_
            color = "orange", size = 2) +
  labs(
    title = "Sepal Length vs Petal Length by Species",
    subtitle = "With Regression Lines and Mean Annotations",
    x = "Sepal Length",
    y = "Petal Length",
    color = "Species"
  ) +
  theme_dark() +
  theme(legend.position = "top")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



ML MODEL- SPLITTING AND CLEANING DATA

First, we check for na or 'null' values in the data set, as we want to make sure that there will be no issues in the Model's ability to accurately read data. We do this by executing function "sum(is.na(iris)). For this data set, we have no 'null' values. Then, we set the seed to 123. Setting the seed ensures that each time this model is run, we can replicate the same results. 123 is a common seed to set, which is why I chose to use it in this model. Finally, we create a Training and Testing set, which will be used to train and test the data. In the TrainingIndex portion of this code, p = .8 refers to the percentage of data that the model will use Train itself and verify trends. On the testing phase of this model, the algorithim will utilize the trends it noticed from the TrainingSet to test its predictions on the 20% of data that was was not used in the TrainingIndex. This helps to ensure that the model can accurately make predictions about new data.

```
sum(is.na(iris))
```

```
## [1] 0
```

```
set.seed(123)

TrainingIndex = createDataPartition(iris$Species, p =.8, list = FALSE)
TrainingSet = iris[TrainingIndex,]
TestingSet = iris[-TrainingIndex,]
```

BUILIDNG THE TRAINING MODEL

Next, we build the training model. To do this, we specify which category we want the model to make predictions about. Since this set has only 1 categorical variable, we have utilized 'Species' variable as the prediction outcome. Additionally, we have no NA values, so we have omitted any action regarding NA values. Finally, we have utilized 'svmPoly' as the method, since its a classification method that works best with smaller data sets. As we mentioned above, the data has some overlapping connections between species. svmPoly compares better than other methods as working with non-linear data, which is why we have utilized it in the training model.

```
Model = train(Species ~., data = TrainingSet,
              method = 'svmPoly',
              na.action = na.omit,
              preProcess= c('scale', 'center'),
              trControl = trainControl(method = 'none'),
              tuneGrid = data.frame(degree = 1, scale = 1, C = 1)
       )
```

BUILDING CROSS-VARIANCE MODEL-

Next, we create a cross-variance model to further increase the accuracy of our data. Essentially, this model will leave out 1 flower in the data alloted to the training matrix. Then, it will try to predict that one flower's species. It will do this to each flower in the data set until each one has been left out of the training prediction once. As each flower gets left out and replaced, the model starts to learn which predictors lead to different species.

```
model.cv = train(Species ~ ., data = TrainingSet,
                 method = "svmPoly",
                 na.action = na.omit,
                 preProcess=c('scale','center'),
                 trControl = trainControl(method = 'cv', number = 10),
                 tuneGrid = data.frame(degree = 1, scale = 1, C =1))
```

APPLYING THE MODEL FOR PREDICTIONS-

In this phase, we actually execute the models we created above. As we can see, the Training model is executed first, followed by the cross variance model. This is because the cross variance model utilized the trends created from the training and testing models and then applies them to the data set. By doing so, the accuracy of the model can be further increased.

```
Model.training = predict(Model, TrainingSet)
Model.testing = predict(Model, TestingSet)
Model.cv = predict(model.cv, TrainingSet)
```

MODEL PREFORMANCE-

after executing the models, we must calculate the accuracy of these models. As we can see, the model had an accuracy of around 99.17 percent for this seed. Additionally, we can see that that the p value is well below 2%, making the key-factors in the model's predictions statistically significant. This means that the probability for Sepal length, width and Petal length and width to be correlated to specific species from chance alone is extremely low.

```
levels(Model.training) <- levels(TrainingSet$Species)
levels(Model.testing) <- levels(TestingSet$Species)
levels(model.cv) <- levels(TrainingSet$Species)

Model.training.confusion = confusionMatrix(Model.training, TrainingSet$Species)
Model.testing.confusion = confusionMatrix(Model.testing, TestingSet$Species)


print(Model.training.confusion)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         40          0         0
##   versicolor      0         39         0
##   virginica       0          1        40
##
## Overall Statistics
##
##                Accuracy : 0.9917
##                  95% CI : (0.9544, 0.9998)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9875
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9750           1.0000
## Specificity                 1.0000            1.0000           0.9875
## Pos Pred Value              1.0000            1.0000           0.9756
## Neg Pred Value              1.0000            0.9877           1.0000
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3250           0.3333
## Detection Prevalence        0.3333            0.3250           0.3417
## Balanced Accuracy           1.0000            0.9875           0.9938
```

```
print(Model.testing.confusion)
```

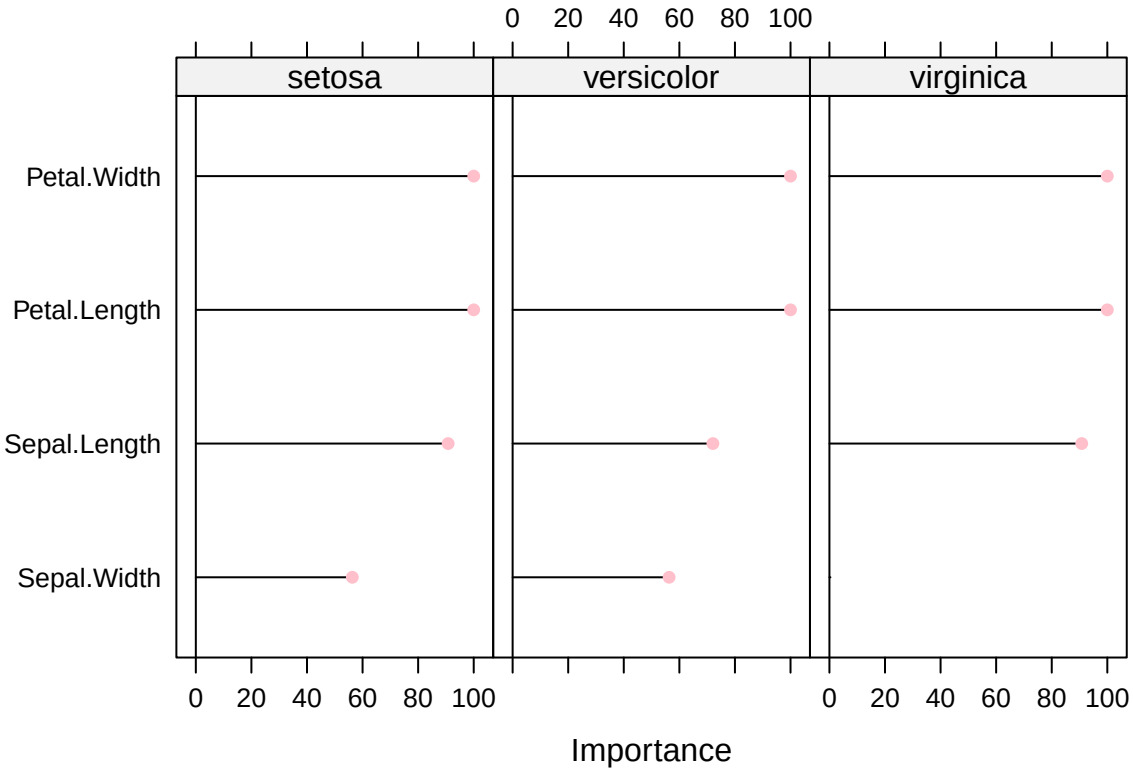
```
## Confusion Matrix and Statistics
```

```
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         10         1
##   virginica       0          0         9
##
## Overall Statistics
##
##                Accuracy : 0.9667
##                  95% CI : (0.8278, 0.9992)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 2.963e-13
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9000
## Specificity                 1.0000            0.9500           1.0000
## Pos Pred Value              1.0000            0.9091           1.0000
## Neg Pred Value              1.0000            1.0000           0.9524
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3333           0.3000
## Detection Prevalence        0.3333            0.3667           0.3000
## Balanced Accuracy           1.0000            0.9750           0.9500
```

PLOTTING THE VARIABLES USED FOR PREDICTION-

finally, we must see exactly which variables contributed most to the model's predictions. As we can see below, we used the varImp function found in the Caret package to plot the importance of each variable in the predictions. As we can see across all three flower species, Petal Width and Length were the most important factors in determining which species a flower belonged to. Additionally, we can see that for Versicolor, Sepal Length wasn't as important in determining the species than for Setosa and Virginica. Finally, we can see that Sepal Length didn't matter at all in determining Virginica flowers. From some of the analysis we did above, we can see why the median Sepal Length didn't affect Versicolor flowers as much as the other species. Because the Versicolor species had a median sepal length of ~5.75, while Virginica had a mean of ~6.5 and Setosa had a median of 5, the variance between it's sepal length and it's sepal length between the other species was less than the average sepal length between Setosa and Virginica species. This higher difference in mean Sepal length made it easier for the model to predict these sepcies based on this variable.

```
Importance = varImp(Model)
plot(Importance, col = 'pink')
```


0 20 40 60 80 100
setosa
versicolor
virginica
Petal.Width
Petal.Length
Sepal.Length
Sepal.Width
0 20 40 60 80 100
0 20 40 60 80 100
Importance