# Iris Final Project

Orel Adani

2023-11-16

## 1 Introduction

Iris, introduced by Ronald Fisher in his 1936 paper The use of multiple measurements in taxonomic problems, contains three plant species (setosa, virginica, versicolor) and four features measured for each observation. These quantify the morphological variation of the iris flower in its three species, all measurements given in centimeters. Ronald measure the sepal and a petal of the flower to the width and length of iris species. Our goal is to create an algorithm that will determine by entering that 4 measurements which species we entered.

## 1.1 Prepare the Data

### 1.1.1 Looking at the data

To understand the data more closely, we will take 2 lines of each species. We can clearly see that some measures has a unique range of numbers and others measures vaguely the same

```
#show the data

data("iris")

colnames(iris)<-c("SL","SW","PL","PW","Species")

rbind(
head(iris,n=2),
head(iris[which(iris$Species=="versicolor"),],n=2),
head(iris[which(iris$Species=="virginica"),],n=2)
)
```

```
##      SL  SW  PL  PW    Species
## 1   5.1 3.5 1.4 0.2     setosa
## 2   4.9 3.0 1.4 0.2     setosa
## 51  7.0 3.2 4.7 1.4 versicolor
## 52  6.4 3.2 4.5 1.5 versicolor
## 101 6.3 3.3 6.0 2.5  virginica
## 102 5.8 2.7 5.1 1.9  virginica
```

By grouping the species and measure the mean max and min for each field, Its very hard to determine which measurement belongs to the right species at the sepal samples.

```
iris %>%
  group_by(Species)  %>% summarize(AVG_SL=mean(SL),MIN_SL=min(SL),MAX_S
L=max(SL),
                                   AVG_SW=mean(SW),MIN_SW=min(SW),MAX_S
W=max(SW)
                                   )
```

```
## # A tibble: 3 × 7
##   Species    AVG_SL MIN_SL MAX_SL AVG_SW MIN_SW MAX_SW
##   <fct>       <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 setosa       5.01    4.3    5.8   3.43    2.3    4.4
## 2 versicolor   5.94    4.9    7     2.77    2      3.4
## 3 virginica    6.59    4.9    7.9   2.97    2.2    3.8
```

```
iris %>%
  group_by(Species)  %>% summarize(AVG_PL=mean(PL),MIN_PL=min(PL),MAX_P
L=max(PL),
                                   AVG_PW=mean(PW),MIN_PW=min(PW),MAX_P
```

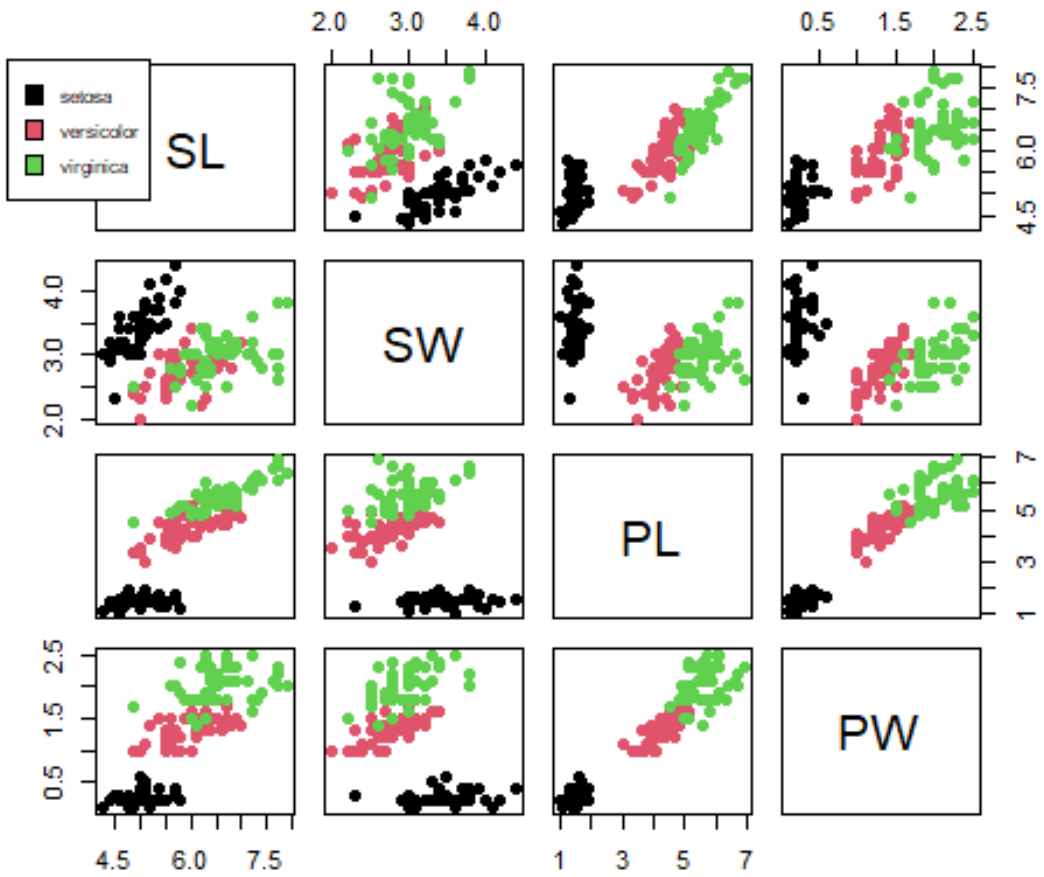
```
W=max(PW)
                                    )
```

```
## # A tibble: 3 × 7
##   Species    AVG_PL MIN_PL MAX_PL AVG_PW MIN_PW MAX_PW
##   <fct>       <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 setosa       1.46    1      1.9  0.246    0.1    0.6
## 2 versicolor   4.26    3      5.1  1.33     1      1.8
## 3 virginica    5.55    4.5    6.9  2.03     1.4    2.5
```

### 1.1.2 Field selection diagnostics

In petal the situation is different, here we can see that each species has a different and unique range. For virginica there are overlapping cases with the versicolor but most of the observations are separated into a clear division. Maybe in the graph it will be more clear to see the difference between the two.

```
#plot all potential predictors
pairs(iris[, 1:4], col = factor(iris$Species), pch = 19, oma=c(2,5,2,2)
, data = iris)
par(xpd = TRUE)
legend("topleft", fill = unique(iris$Species),cex=0.5, legend = c( leve
ls(iris$Species)))
```



We found the appropriate fields to help us distinguish between the species. Receiving the relevant fields will help the algorithm to run faster and yet will not detract from the findings being accurate

### 1.1.3 scaling the data

Scaling is a technique for comparing data that isn't measured in the same way. In the machine learning algorithms if the values of the features are closer to each other.There is a chance which the algorithm get trained well and faster but when the data points or features values have high differences with each other, it will take more time to understand the data and the accuracy may be lower.

```
#choose the right predictors

iris<-iris %>%  select("PL","PW","Species")


#scaling the data

maxval<-apply(iris[,1:2],2, max)
minval<-apply(iris[,1:2],2, min)

iris_scale<-as.data.frame(scale(iris[,1:2],center=minval,scale=(maxval-
minval)))
iris_data_round<-cbind(round(iris_scale,4),Species=iris$Species)
iris_data<-cbind(iris_scale,Species=iris$Species)

head(iris_data_round)
```

```
##       PL     PW Species
## 1 0.0678 0.0417  setosa
## 2 0.0678 0.0417  setosa
## 3 0.0508 0.0417  setosa
## 4 0.0847 0.0417  setosa
## 5 0.0678 0.0417  setosa
## 6 0.1186 0.1250  setosa
```

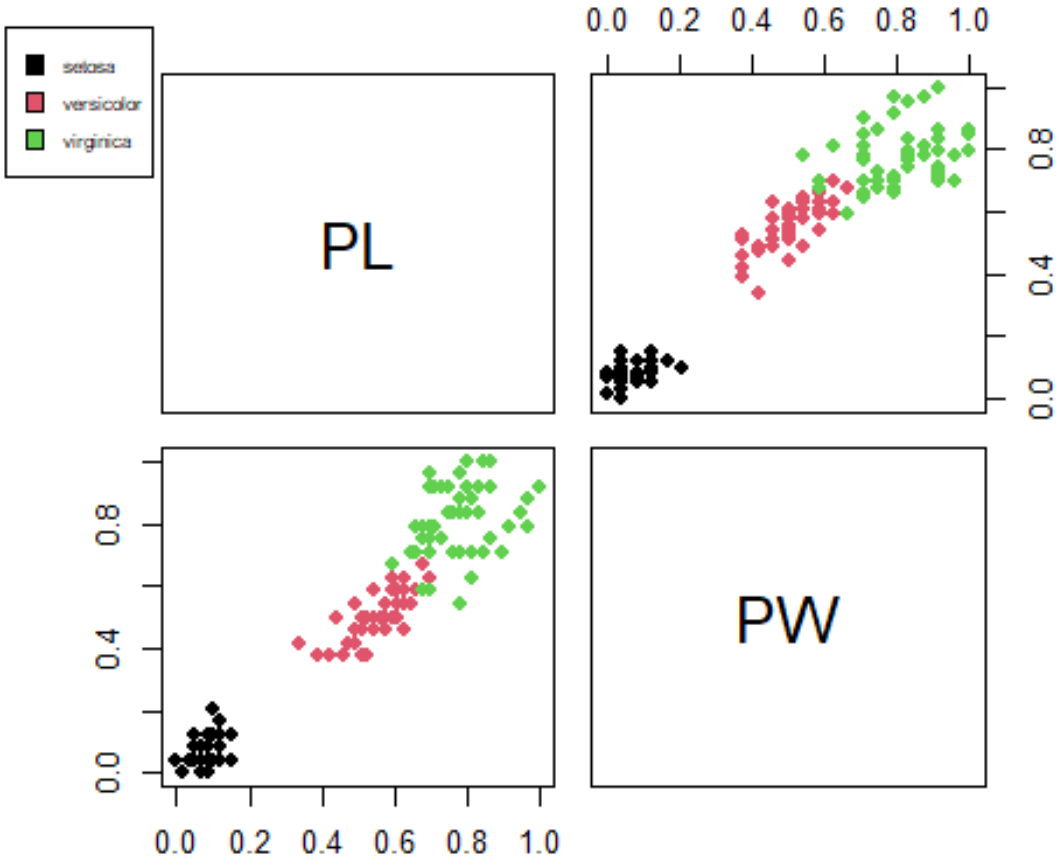
```
rbind(
  head(iris_data_round,n=2),
  head(iris_data_round[which(iris_data_round$Species=="versicolor"),],n
=2),
  head(iris_data_round[which(iris_data_round$Species=="virginica"),],n=
2)
)
```

```
##         PL     PW    Species
## 1   0.0678 0.0417     setosa
## 2   0.0678 0.0417     setosa
## 51  0.6271 0.5417 versicolor
## 52  0.5932 0.5833 versicolor
```

```
## 101 0.8475 1.0000  virginica
## 102 0.6949 0.7500  virginica
```

By subtracting the values of each column by the matching "center" value from the argument we reserve the place of the value from his other values in the field and shrinking the range from 0-1. As we can see, the graphs remain the same but the axes have changed.

```
pairs(iris_data[, 1:2], col = factor(iris_data$Species), pch = 19, oma=
c(2,5,2,2), data = iris_data)
par(xpd = TRUE)
legend("topleft", fill = unique(iris_data$Species),cex=0.5, legend = c(
levels(iris_data$Species)))
```




## 1.2 Creating the "Test" and "Training" sets

Partitioning the data into 2 sets: 70% of the data will be in the train set, and 30% in the Test set. We take a large sample of test because if we divide them to 3 species we left with 15 samples each. Taking less will leave us with poor number of test set.

```r
#Partitioning the data to train and test

set.seed(1, sample.kind="Rounding") # if using R 3.6 or later

index_train <- createDataPartition(iris_data$Species, p=0.7, list = FAL
SE)

test <-  iris_data[-index_train,]
train <- iris_data[index_train,]

table(train$Species)

##
##     setosa versicolor  virginica
##         35         35         35
```

Getting 45 samples of test and 105 in the train set. each species will have equal number of samples.

## 2 Comparison between two models

we going to test two models explaining their method and see how well they succeed to predict the species. Each has its advantages and disadvantages in such way that the performance of the algorithm is affected by them.

- **Decision Trees**
- **Artificial Neural Network**

At the end of the analysis, we will be able to compare the different results and determine which one is more accurate in our diagnosis based on this database (different databases will need different methodologies)

### 2.1 Decision Trees

Decision Tree is a supervised machine learning algorithm which can be used to perform both classification and regression on complex data sets. Hence, it works for both continuous and categorical variables. Important basic tree Terminology is that the Root node represents an entire population or data set which gets divided into two or more pure sets (also known as homogeneous steps). It always contains a single input variable until we Leaf with terminal node that do not split further and contains the output variable.

We are partitioning the predictor space into $J$ non-overlapping regions:

$R1$ - Setosa

$R2$ - Versicolor

$R3$ - Virginica

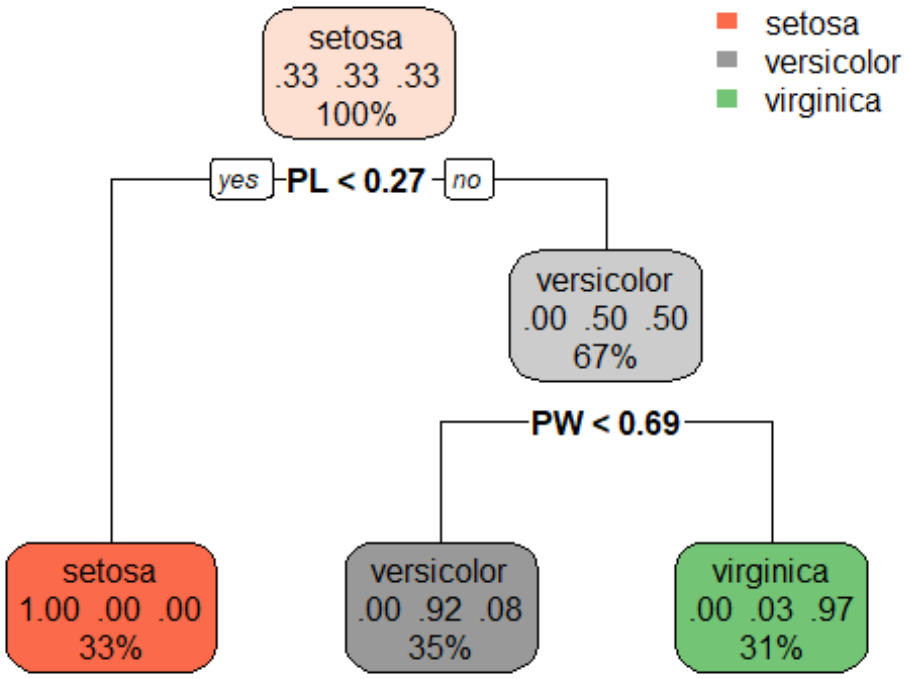
Then for any predictor that falls within region we estimate $f(x)$ minimizes the residual sum of squares (RSS):

$$RSS = \sum_{i:xiR1(j,s)} (\hat{y}_i - y_{R1})^2 + \sum_{i:xiR2(j,s)} (\hat{y}_i - y_{R2})^2 + \sum_{i:xiR3(j,s)} (\hat{y}_i - y_{R3})^2$$

```
#testing classification tree and plot it

train_rpart <- rpart(Species~., data = train)

rpart.plot(train_rpart)
```




To understand more deeply the intent of the model we draw **straight line** to each veritable that cut our set of observations into different segments.

### 2.1.1 compute accuracy in rpart train

```
# compute accuracy in train

Predict_rpart_train <-  predict(train_rpart, newdata = train, type = "c
lass")

table(train$Species, Predict_rpart_train)

##             Predict_rpart_train
##              setosa versicolor virginica
##   setosa         35          0         0
##   versicolor      0         34         1
##   virginica       0          3        32

accuracy_rpart_train<- percent(mean(train$Species==Predict_rpart_train)
, accuracy = 0.1)
error_rate_rpart_train<-percent(mean(train$Species!=Predict_rpart_train
), accuracy = 0.1)
```

To summarize the results: The accuracy of the decision tree saw fit to use our two variables that split the species and decided which group belonged to which species.The algorithm at the train set accuracy for that model is **96.2%** Hence, it's error rate stands at **3.8%** .

### 2.1.2 compute accuracy in rpart test

With such promising findings we will apply the algorithm to the test set:

```
# compute accuracy in test

Predict_rpart_tests <-  predict(train_rpart, newdata = test, type = "cl
ass")

table(test$Species, Predict_rpart_tests)

##             Predict_rpart_tests
##              setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         15         0
##   virginica       0          2        13

accuracy_rpart_test<- percent(mean(test$Species==Predict_rpart_tests),
accuracy = 0.1)
error_rate_rpart_test<-percent(mean(test$Species!=Predict_rpart_tests),
accuracy = 0.1)
```

The algorithm results at the test set accuracy for that model is **95.6%** Hence, it's error rate stands at **4.4%** .

As satisfying as the results may be,

**is it possible to create a more accurate model than this?**

## 2.2 Artificial Neural Networks

Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

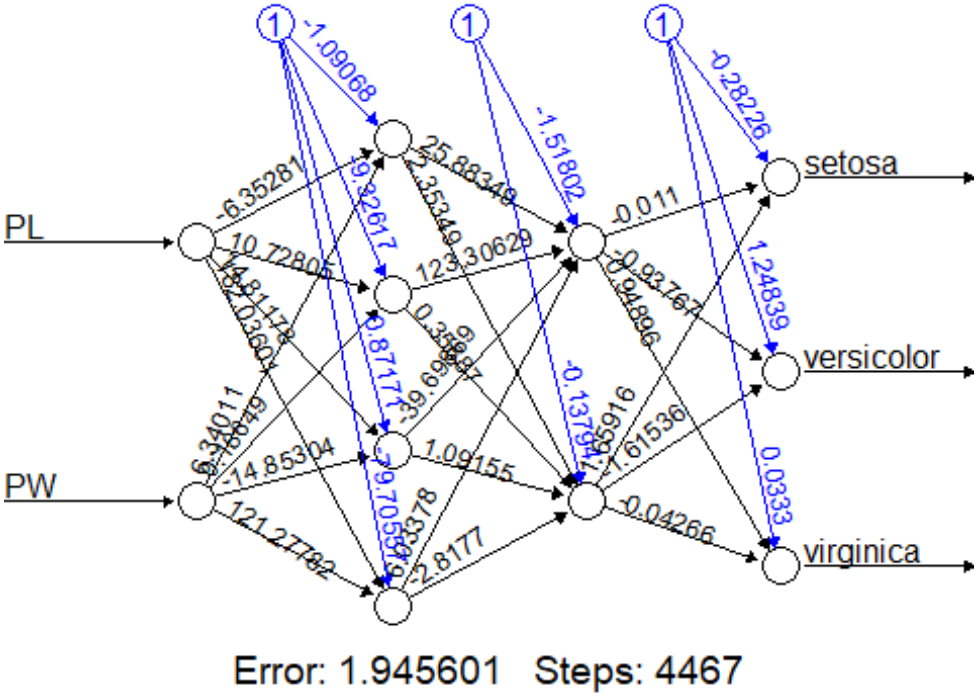
Artificial neural networks (ANNs) are comprised of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold.The output layer nodes are dependent on their immediately preceding hidden layers and those nodes are further derived from the input variables. in each neuron formula being uses

$$Neron_x = \sum_i (w_i x_i) + b$$

and the answer move to the next layer neuron. it will be easier to see in a graph

```
#creating a neural net work on train_set to predict the Species
ANN<-neuralnet:: neuralnet(Species  ~ ., train , hidden =c(4,2))
```

```
plot(ANN,rep = "best")
```




By looking at Our model we see minor weight for each $b_i$ which indicates a greater weight to the neurons located in each layer.

To understand more deeply the intent of the model we draw **line that curving in strategic places** to each veritable and cut our set of observations into different segments.

### 2.2.1 compute accuracy in ANN train

```
# compute accuracy in train

pred_set = neuralnet:: compute(ANN,train[,c(1,2,3)])

pred_df <- data_frame()

n=c(length(train$Species))

for (i in 1:n) {
  pred_df<- rbind(pred_df, which.max(pred_set$net.result[i,]))
}

pred_df$X1L<- gsub(1,"setosa", pred_df$X1L)
pred_df$X1L<- gsub(2,"versicolor", pred_df$X1L)
pred_df$X1L<- gsub(3,"virginica", pred_df$X1L)
```

```
prediction <- pred_df$X1L
reference <- train$Species

accuracy_ann_train<-percent(mean(prediction == reference), accuracy = 0
.1)
error_ann_train<-percent(mean(prediction != reference), accuracy = 0.1)

table(prediction, reference)

##             reference
## prediction   setosa versicolor virginica
##   setosa         35          0         0
##   versicolor      0         34         1
##   virginica       0          1        34
```

To summarize the results: The accuracy of the artificial neural networks saw fit to use our two variables by entered them in the chain of neurons and multiplied by the weights and gives as the final output layer to determine which observation belonged to which species.The algorithm at the train set accuracy for that model is **98.1%** Hence, it's error rate stands at **1.9%** .

### 2.2.2 compute accuracy in ANN test

With such promising findings we will apply the algorithm to the test set:

```
# compute accuracy in test

perd_test<- predict(ANN,test,type="class")
reference_test <-test$Species

validation_df <-data.frame(x=apply(perd_test,1,which.max))

validation_df<-ifelse( validation_df$x==1,"setosa",
                       ifelse(validation_df$x==2,"versicolor",
                              "virginica"))

table(reference_test,validation_df)

##                validation_df
## reference_test setosa versicolor virginica
##     setosa         15          0         0
##     versicolor      0         15         0
##     virginica       0          1        14

accuracy_ann_test<-percent(mean(reference_test==validation_df) , accura
cy = 0.1)
error_ann_test<-percent(mean(reference_test!=validation_df) , accuracy
= 0.1)
```

The algorithm results at the test set accuracy for that model is **97.8%** Hence, it's error rate stands at **2.2%** .

## 3 Results Comparison between Models

To determine which model produces more accurate results,we can't rely on just one sample. Preparing a loop that will create a table in which all the results of the two models will store. We will create 30 rows where each row is a random sample of the database by changing the seed set that responsible for mixing the data every round a loop starts.

This process takes time because each loop generates all the models from scratch. But the findings shed light on the viability of the preferred model. (It's worth being patient)

```
#check the the best method

N_seeds<- c(1,12,123,1234,12345,123456,2089, 2676, 4831, 4261,132,2344,
2334,3366,5465,
    380,99,876,594,1998,35,45,67,878,23,8766,3401,31,84,96)

models_results <- data_frame()
models_results_final <- data_frame()

for (i in 1:length(N_seeds)) {
  set.seed(N_seeds[i]) # if using R 3.6 or later
  #create data
  index_train <- createDataPartition(iris_data$Species, p=0.7, list = F
ALSE)
  test <-  iris_data[-index_train,]
  train <- iris_data[index_train,]
  train_rpart <- rpart(Species~., data = train)
  #rpart
  Predict_rpart_tests <-  predict(train_rpart, newdata = test, type = "
class")
  accuracy_rpart_test<- mean(test$Species==Predict_rpart_tests)
  error_rate_rpart_test<-mean(test$Species!=Predict_rpart_tests)
  #NNA
  ANN<-neuralnet:: neuralnet(Species  ~ ., train , hidden =c(4,2))
  perd_test<- predict(ANN,test,type="class")
  reference_test <-test$Species
  validation_df <-data.frame(x=apply(perd_test,1,which.max))
  validation_df<-ifelse( validation_df$x==1,"setosa",
                         ifelse(validation_df$x==2,"versicolor",
                                "virginica"))
  table(reference_test,validation_df)
```

```
  accuracy_ann_test<-mean(reference_test==validation_df)
  error_ann_test<-mean(reference_test!=validation_df)
  #summery
   models_results <- bind_rows(tibble(N_seeds ,accuracy_rpart=accuracy_
rpart_test,error_rate_rpart=error_rate_rpart_test,
                                      accuracy_ann=accuracy_ann_test,err
or_ann=error_ann_test,
                                      diff_accuracy=accuracy_ann-accurac
y_rpart
  ))
  models_results_final<-rbind(models_results_final,models_results[i,])
}
summary_count<-models_results_final%>%summarise(count_0=length(which(di
ff_accuracy==0)),
                                                count_p=length(which(di
ff_accuracy>0)),
                                                count_n=length(which(di
ff_accuracy<0))
                                                )

models_results_final<-models_results_final %>% mutate(
  accuracy_rpart= percent(accuracy_rpart, accuracy = 0.1),
  error_rate_rpart= percent(error_rate_rpart , accuracy = 0.1),
  accuracy_ann= percent(accuracy_ann , accuracy = 0.1),
  error_ann= percent(error_ann , accuracy = 0.1),
  diff_accuracy= percent(diff_accuracy, accuracy = 0.1),
  )

head(models_results_final)
```

```
## # A tibble: 6 × 6
##   N_seeds accuracy_rpart error_rate_rpart accuracy_ann error_ann dif
f_accuracy
##     <dbl> <chr>          <chr>            <chr>        <chr>     <ch
r>
## 1       1 95.6%          4.4%             97.8%        2.2%      2.2
%
## 2      12 97.8%          2.2%             97.8%        2.2%      0.0
%
## 3     123 93.3%          6.7%             93.3%        6.7%      0.0
%
## 4    1234 93.3%          6.7%             95.6%        4.4%      2.2
%
## 5   12345 97.8%          2.2%             95.6%        4.4%      -2.
2%
## 6  123456 91.1%          8.9%             93.3%        6.7%      2.2
%
```

```
summary_count
```

```
## # A tibble: 1 × 3
##   count_0 count_p count_n
##     <int>   <int>   <int>
## 1      11      17       2
```

## 4 Conclusion

The results obtained clearly showed that ANN gave a better success rate for predicting the species. As it got **17** out of **30** observations. This makes sense because sometimes the curvature of the regression line can contain more observations than a limited straight line. In the existing database it was possible to see that the Decision Tree gave respectable success rates and we could be satisfied with this model for predicting the species.But it got **2** out of **30** observations.So, there is no doubt that we had greater success with the ANN model.