403_project

ZIHAN WANG

10/04/2021

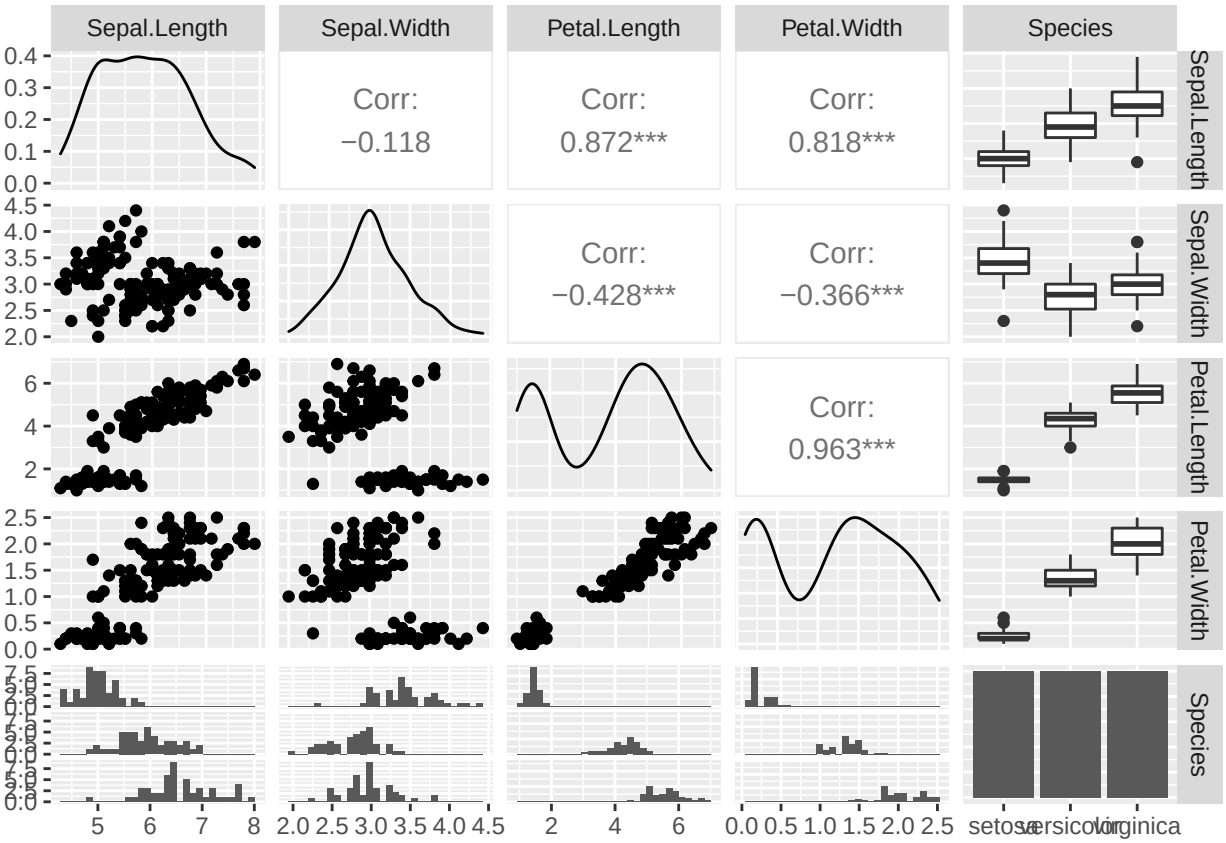
```
library(dplyr)
library(ggplot2)
library(GGally)
library(datasets)
library(stats)
library(ggfortify)
library(lfda)
library("caret")
```

Load Data

```
data(iris)
```

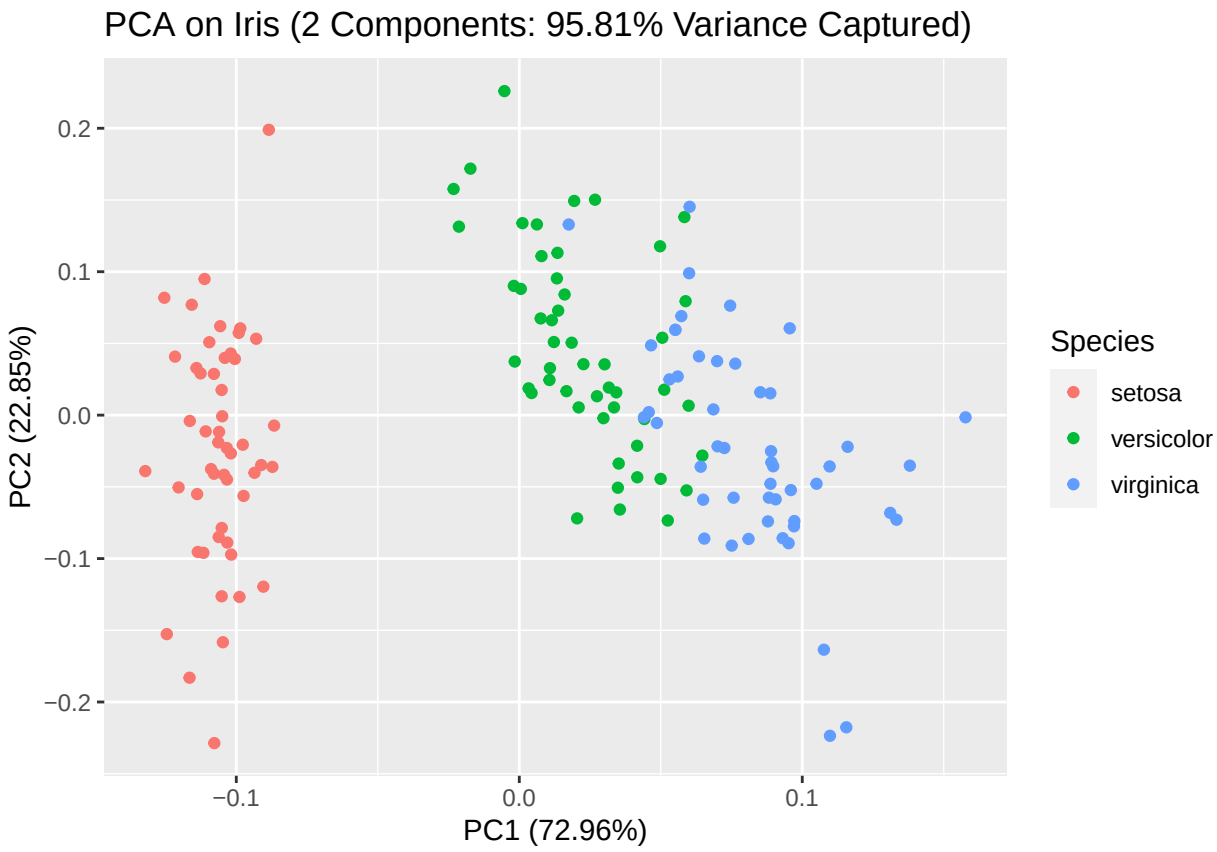
Generalized Pairs Plot on Iris

```
# plot(iris)
ggpairs(iris)
```

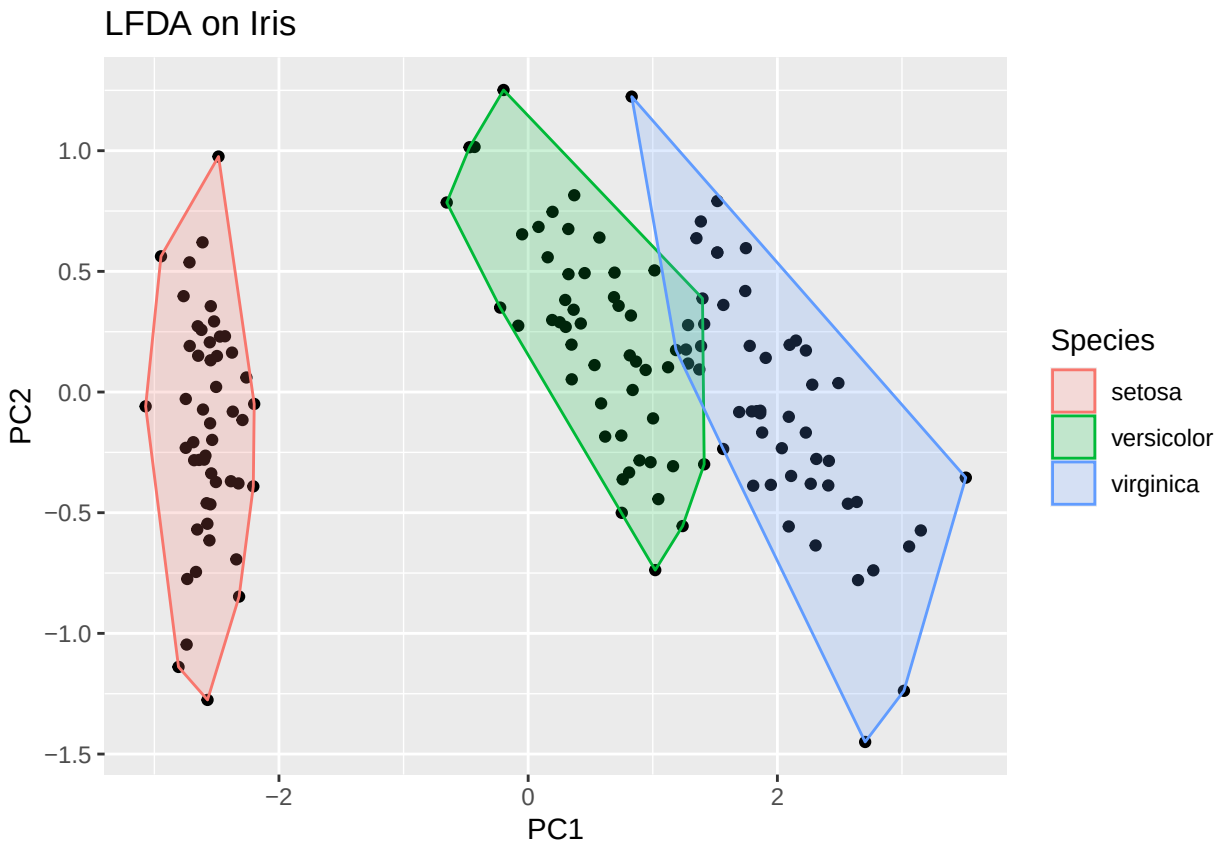


Dimension Reduction Techniques on Iris

```
df = iris[1:4]
pca_res = prcomp(df, scale. = TRUE)

autoplot(pca_res, data = iris, colour = 'Species') +
  ggtitle("PCA on Iris (2 Components: 95.81% Variance Captured)")
```



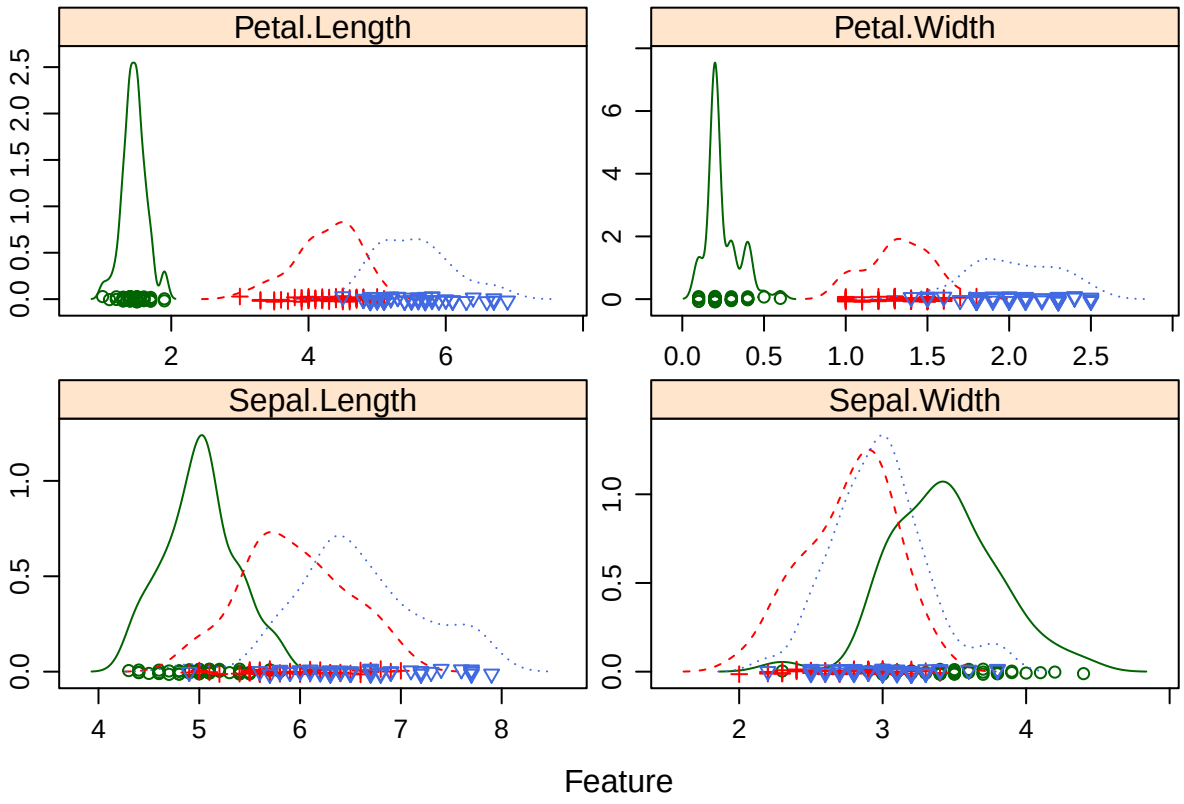

```
model = lfda(iris[-5], iris[, 5], r = 3, metric="plain")
autoplot(model, data = iris, frame = TRUE, frame.colour = 'Species') + ggtitle("LFDA on Iris")
```



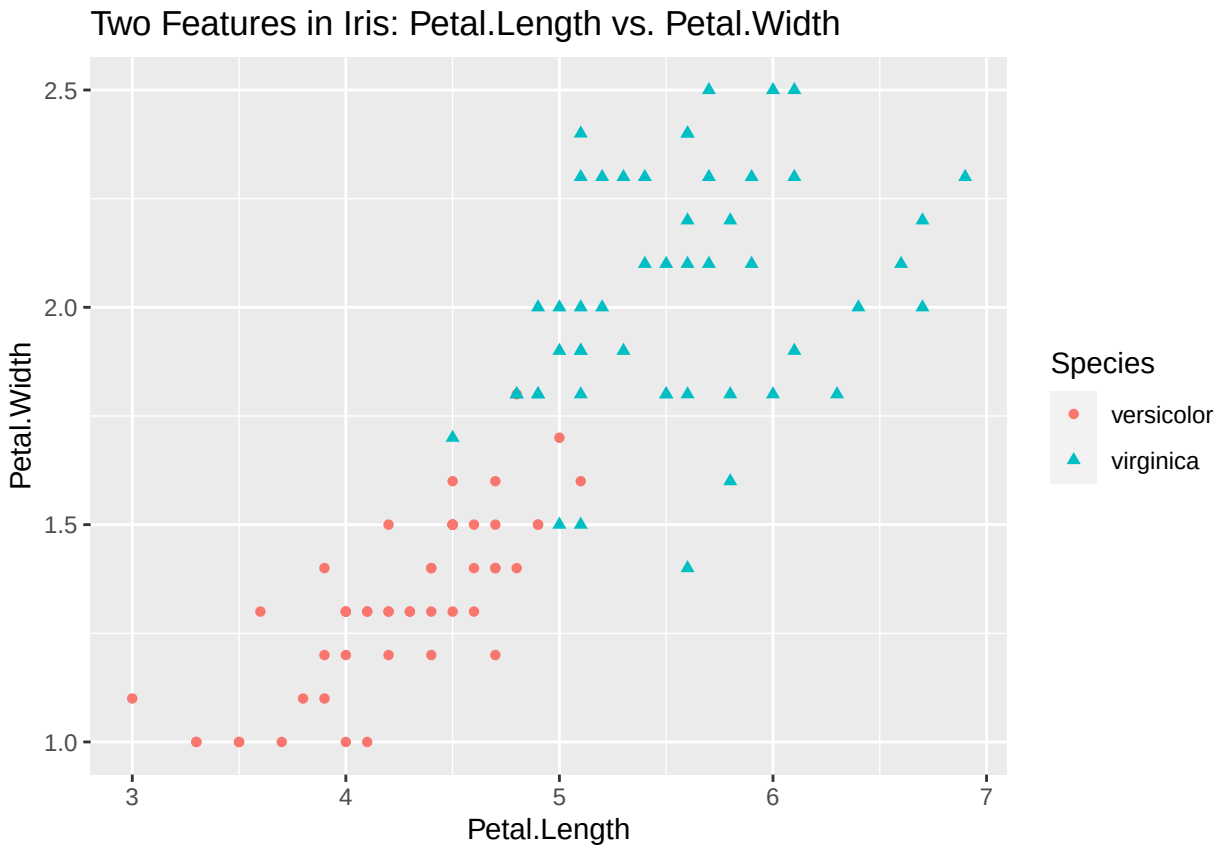
Variable Selection

```
iris_subset = select(iris, Petal.Length, Petal.Width, Species)
iris_subset = iris_subset[iris_subset$Species != "setosa", ]
```

```
iris_x = iris[, 1:4]
iris_y = iris[, 5]
trellis.par.set(theme = col.whitebg(), warn = FALSE)
scales <- list(x=list(relation="free"), y=list(relation="free"))
featurePlot(x=iris_x, y=iris_y, plot="density", scales=scales)
```



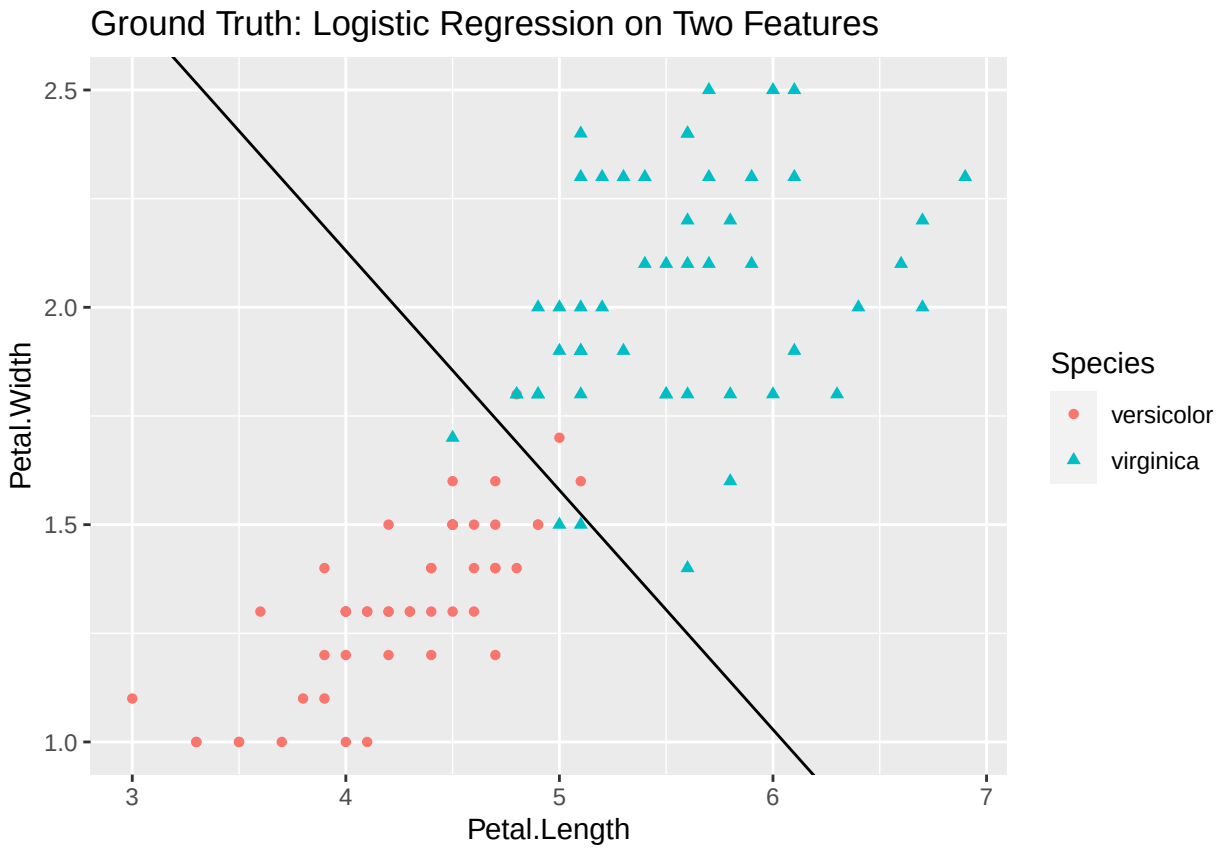

```
ggplot(iris_subset, aes(x=Petal.Length, y=Petal.Width,
                        , shape=Species, color=Species)) + geom_point()+
                    ggtitle("Two Features in Iris: Petal.Length vs. Petal.Width")
```



Build the Ground Truth Model

```
glfit = glm(Species ~ ., data = iris_subset, family = binomial(logit))
```

```
slope = coef(glfit)[2]/(-coef(glfit)[3])
intercept = coef(glfit)[1]/(-coef(glfit)[3])

ggplot(iris_subset, aes(x=Petal.Length, y=Petal.Width,
        , shape=Species, color=Species)) + geom_point() +
        geom_abline(slope = slope, intercept = intercept) +
        ggtitle("Ground Truth: Logistic Regression on Two Features")
```




```
truth_slope = slope
truth_intercept = intercept
```

Split to train and test sets

```
iris_subset$Species = droplevels(iris_subset$Species)
levels(iris_subset$Species) = c(0, 1)

smp_size = floor(0.8 * nrow(iris_subset))
## set the seed to make your partition reproducible
set.seed(123)
train_ind = sample(seq_len(nrow(iris_subset)), size = smp_size)

train = iris_subset[train_ind, ]
test = iris_subset[-train_ind, ]
```

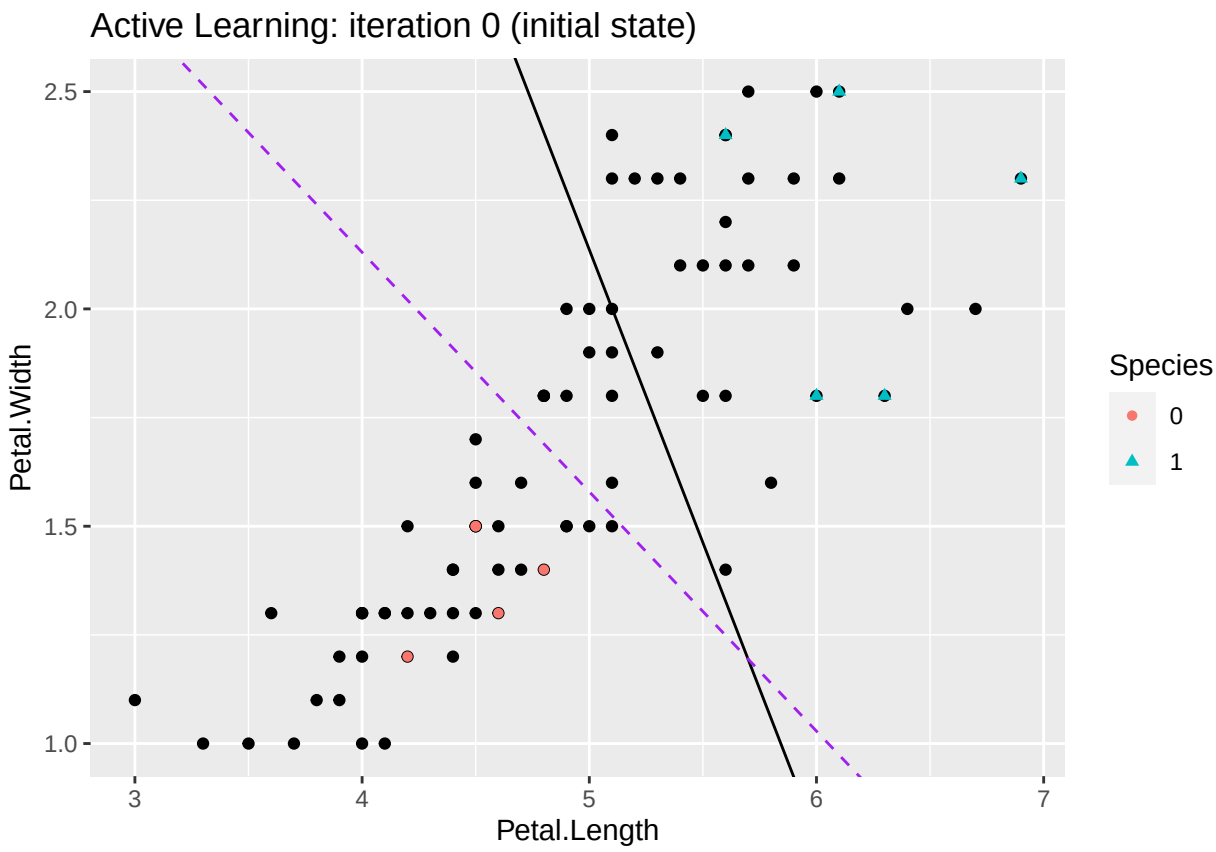
Split train to initial labeled and pool (unlabeled)

```
init_ind = sample(seq_len(nrow(train)), size = 10)

train_init = train[init_ind, ]
train_pool = train[-init_ind, ]
```

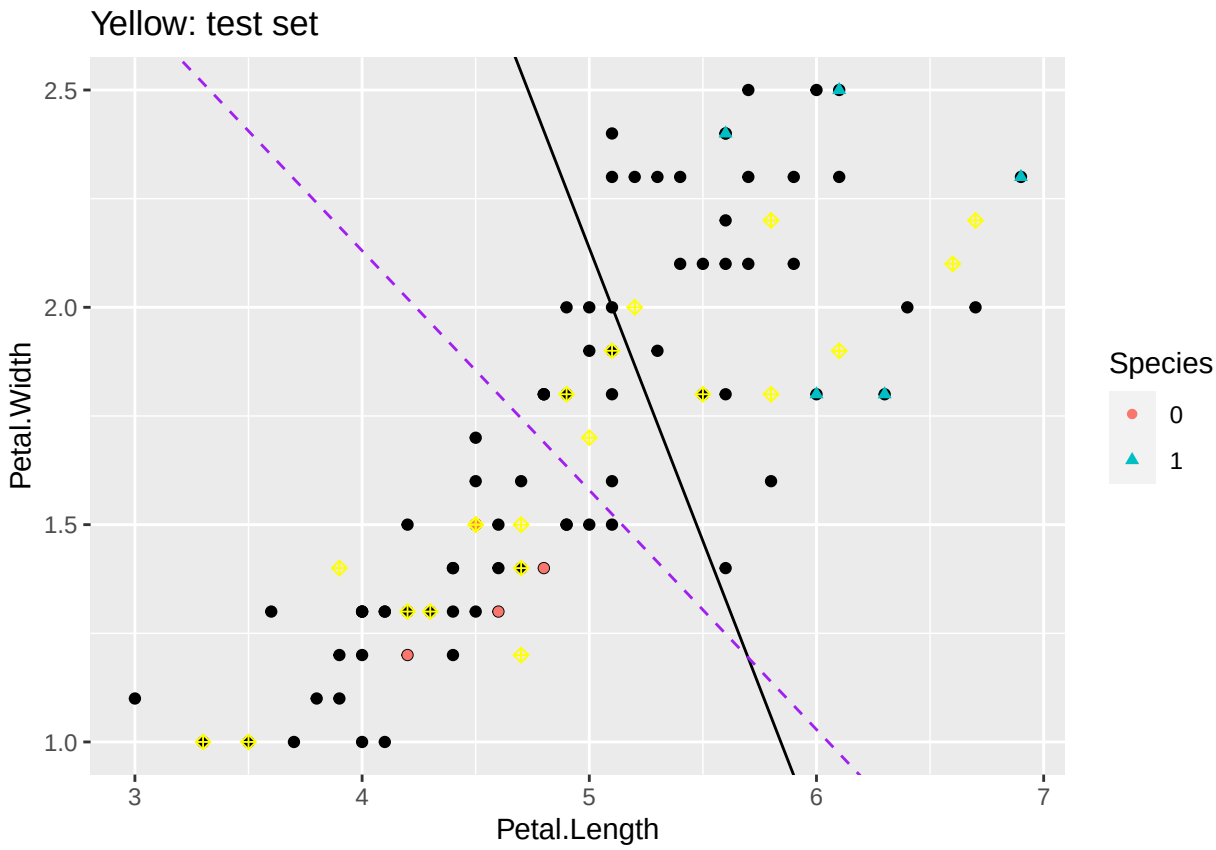
Active Learning: initial state

```
glfit_init = glm(Species ~ ., data = train_init, family = binomial(logit))
```

```
slope_init = coef(glfit_init)[2]/(-coef(glfit_init)[3])
intercept_init = coef(glfit_init)[1]/(-coef(glfit_init)[3])

ggplot() +
geom_point(data = train, aes(x=Petal.Length, y=Petal.Width)) +
geom_abline(slope = slope_init, intercept = intercept_init) +
geom_point(data = train_init,aes(x=Petal.Length, y=Petal.Width , shape=Species, color=Species)) +
  ggtitle("Active Learning: iteration 0 (initial state)") +
geom_abline(slope = truth_slope, intercept = truth_intercept,
            linetype = "dashed", color = "purple")
```




```
ggplot() +
geom_point(data = train, aes(x=Petal.Length, y=Petal.Width)) +
geom_abline(slope = slope_init, intercept = intercept_init) +
geom_point(data = train_init,aes(x=Petal.Length, y=Petal.Width , shape=Species, color=Species)) +
geom_point(data = test, aes(x=Petal.Length, y=Petal.Width), shape = 9, color = "yellow") +
    ggtitle("Yellow: test set") + geom_abline(slope = truth_slope, intercept = truth_intercept, linetyp
```



Active Learning Querying Techniques: Uncertainty Sampling

Distance Functions

```
dist2d = function(a,b,c) {
   v1 = b - c
   v2 = a - b
   m = cbind(v1,v2)
   d = abs(det(m))/sqrt(sum(v1*v1))

   return(d)
}

return_two_points_in_a_line = function(slope, intercept){
  x1 = 5
  x2 = 6
  y1 = slope*x1 + intercept
  y2 = slope*x2 + intercept
  # plot(c(x1,x2), c(y1,y2), xlim = c(3,7))
  point1 = c(x1, y1)
  point2 = c(x2, y2)

  return(list(point1 = point1, point2 = point2))
}
```

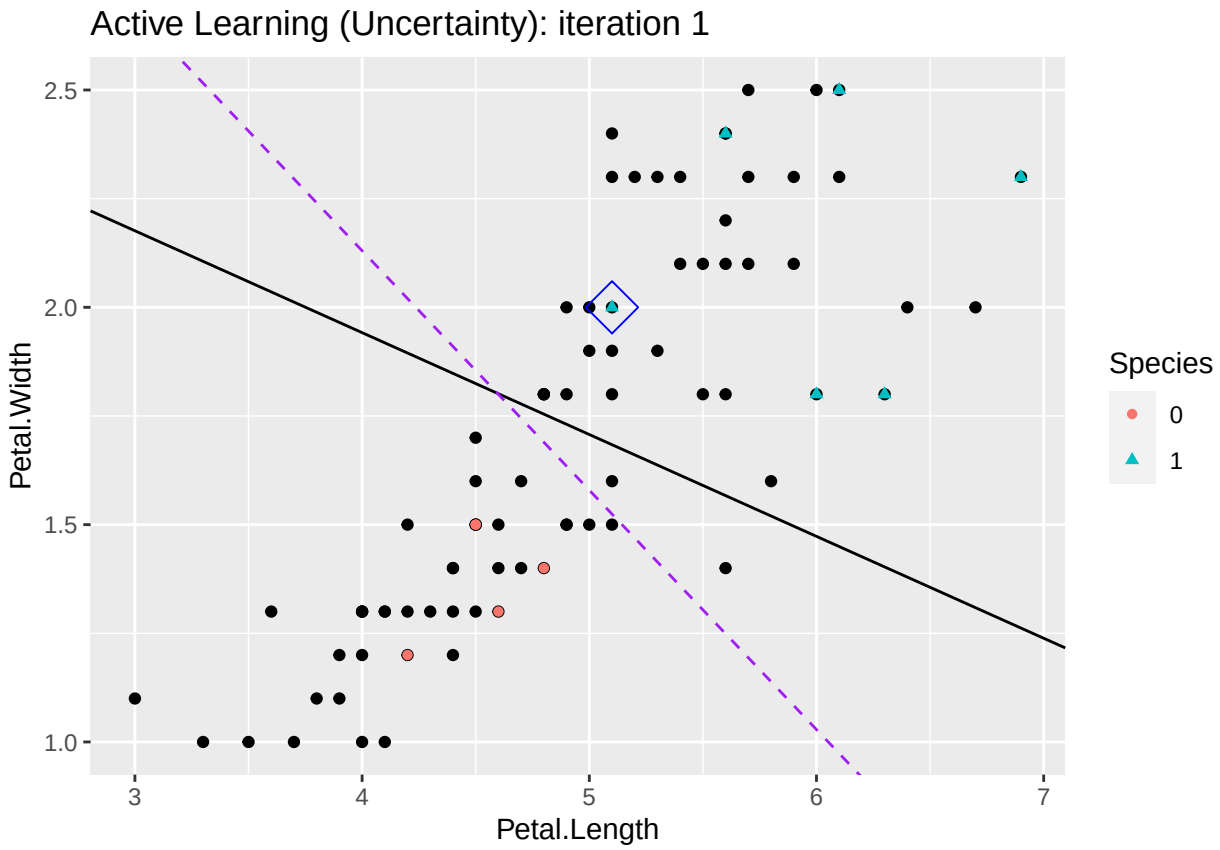
Select the point with "least confidence"

```
two_points = return_two_points_in_a_line(slope_init, intercept_init)

train_pool$dist = NA
for(i in 1:nrow(train_pool)){
  point = c(train_pool$Petal.Length[i], train_pool$Petal.Width[i])
  train_pool$dist[i] = dist2d(point, two_points$point1, two_points$point2)
}
query_point_index = which.min(train_pool$dist)

query_point = train_pool[query_point_index, c(1,2,3)]

train_pool = train_pool[-query_point_index,]

train_init_add = rbind(train_init, query_point)
```

Active Learning: iteration 1

```
glfit_init_add = glm(Species ~ ., data = train_init_add, family = binomial(logit))
```

```
slope_init_add = coef(glfit_init_add)[2]/(-coef(glfit_init_add)[3])
intercept_init_add = coef(glfit_init_add)[1]/(-coef(glfit_init_add)[3])

ggplot() +
geom_point(data = train, aes(x=Petal.Length, y=Petal.Width)) +
geom_abline(slope = slope_init_add, intercept = intercept_init_add) +
geom_point(data = train_init_add,aes(x=Petal.Length, y=Petal.Width , shape=Species, color=Species)) +
geom_point(data = query_point, aes(x=Petal.Length, y=Petal.Width),
           size = 7, shape = 23, color = "blue") +
  ggtitle("Active Learning (Uncertainty): iteration 1")+
  geom_abline(slope = truth_slope, intercept = truth_intercept, linetype = "dashed", color = "purple")
```



Iteration 1: Evaluation

Confusion Matrix

```
predicted_prob = predict(glfit_init, test, type = "response")
predicted_classes = ifelse(predicted_prob > 0.5, 1, 0)
table(test$Species, predicted_classes, dnn = c("Obs", "Pred"))
```

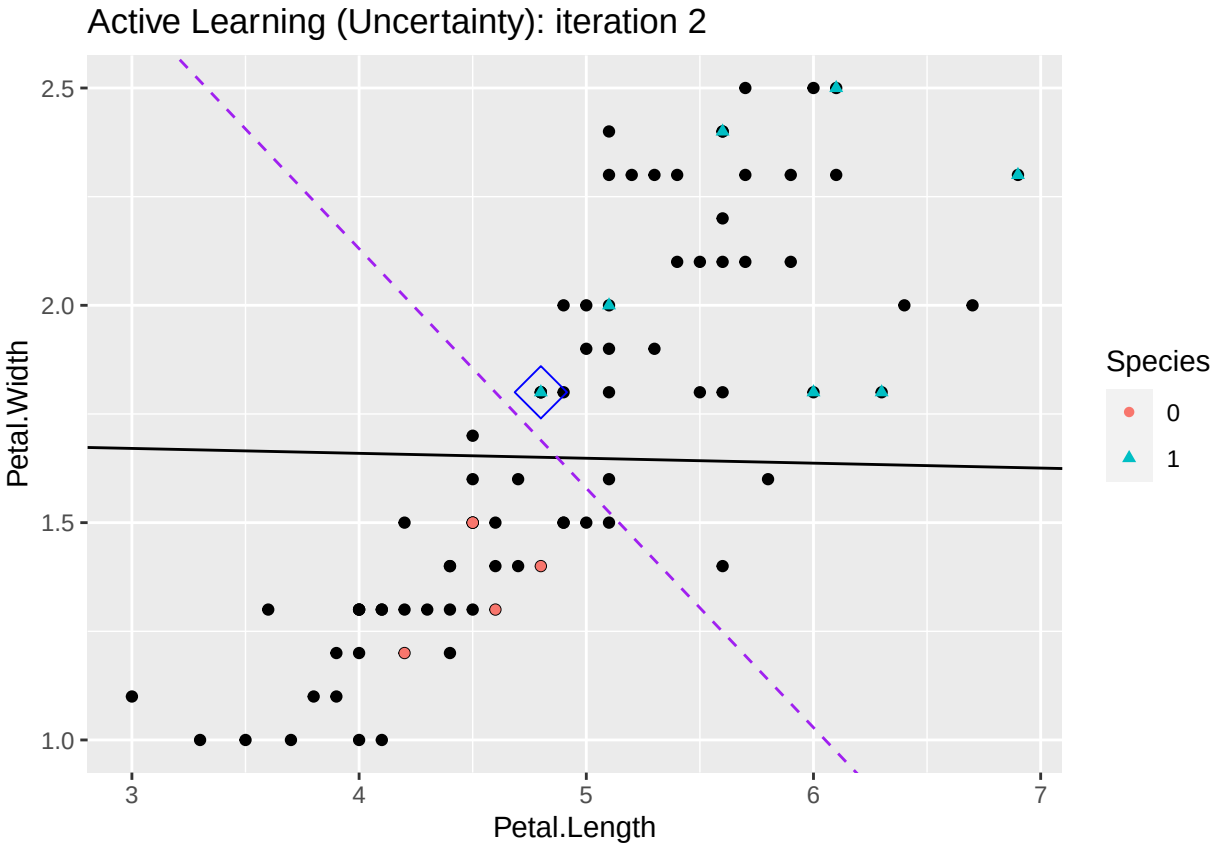
```
##    Pred
## Obs  0  1
##   0 11  0
##   1  2  7
```

Active Learning Process

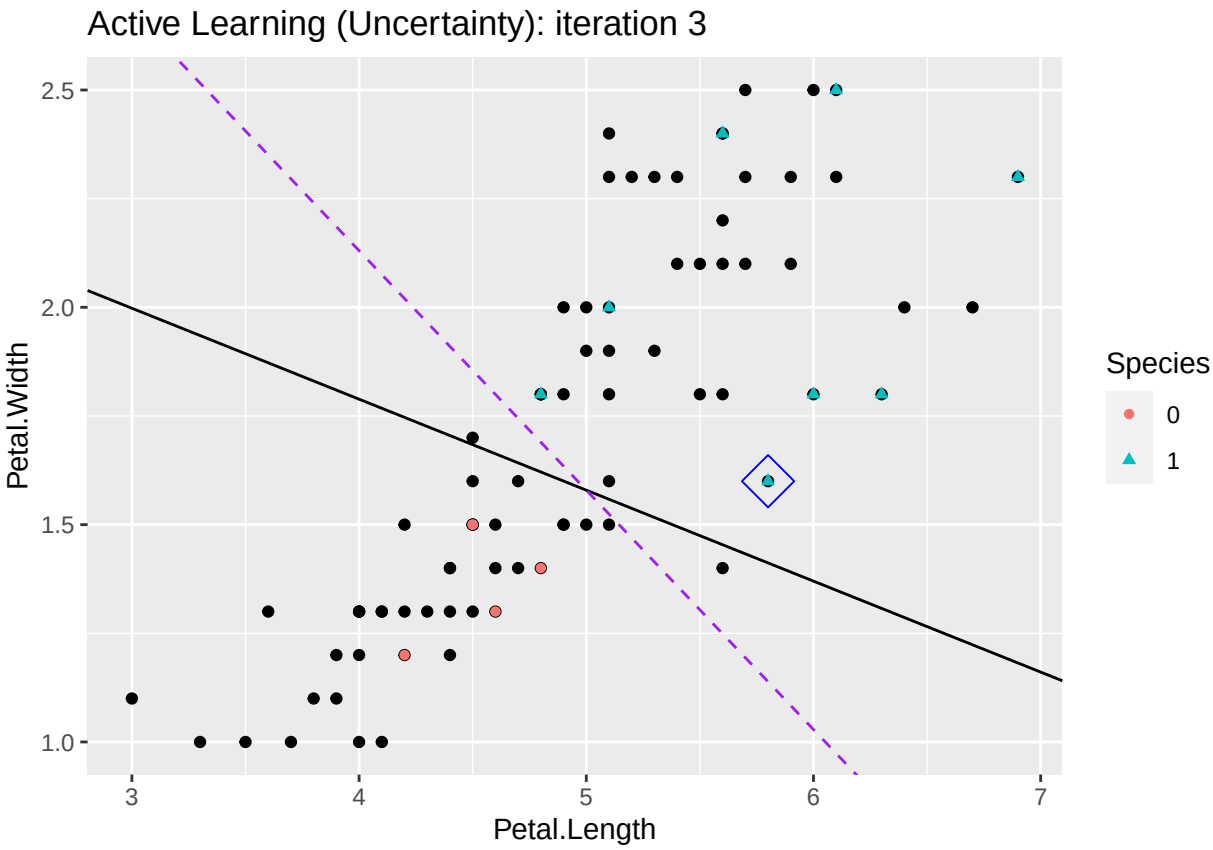
```
for (iter in 2:30){

  two_points = return_two_points_in_a_line(slope_init_add, intercept_init_add)

  # update the distance for points to the new logistic regression line
  train_pool$dist = NA
  for(i in 1:nrow(train_pool)){
    point = c(train_pool$Petal.Length[i], train_pool$Petal.Width[i])
    train_pool$dist[i] = dist2d(point, two_points$point1, two_points$point2)
  }

  # query a new point
```

```
  query_point_index = which.min(train_pool$dist)
  query_point = train_pool[query_point_index, c(1,2,3)]

  train_pool = train_pool[-query_point_index,]
  train_init_add = rbind(train_init_add, query_point)

  # re-train a new model
  glfit_init_add = glm(Species ~ ., data = train_init_add, family = binomial(logit))

  slope_init_add = coef(glfit_init_add)[2]/(-coef(glfit_init_add)[3])
  intercept_init_add = coef(glfit_init_add)[1]/(-coef(glfit_init_add)[3])

  curr_plot = ggplot() +
    geom_point(data = train, aes(x=Petal.Length, y=Petal.Width)) +
    geom_abline(slope = slope_init_add, intercept = intercept_init_add) +
    geom_point(data = train_init_add,aes(x=Petal.Length, y=Petal.Width , shape=Species, color=Species))
    geom_point(data = query_point, aes(x=Petal.Length, y=Petal.Width),
               size = 7, shape = 23, color = "blue") +
        ggtitle(paste0("Active Learning (Uncertainty): iteration ", iter)) +
    geom_abline(slope = truth_slope, intercept = truth_intercept, linetype = "dashed", color = "purple"

  if(iter<=10 | iter == 30){
    ggsave(paste0("AL_process_UC_", iter, ".png"), width = 7, height = 4.33)
    print(paste0("done in iter ", iter))
  }

  if(iter%in%c(2,3,10,30)){
    print(curr_plot)
  }
}
```

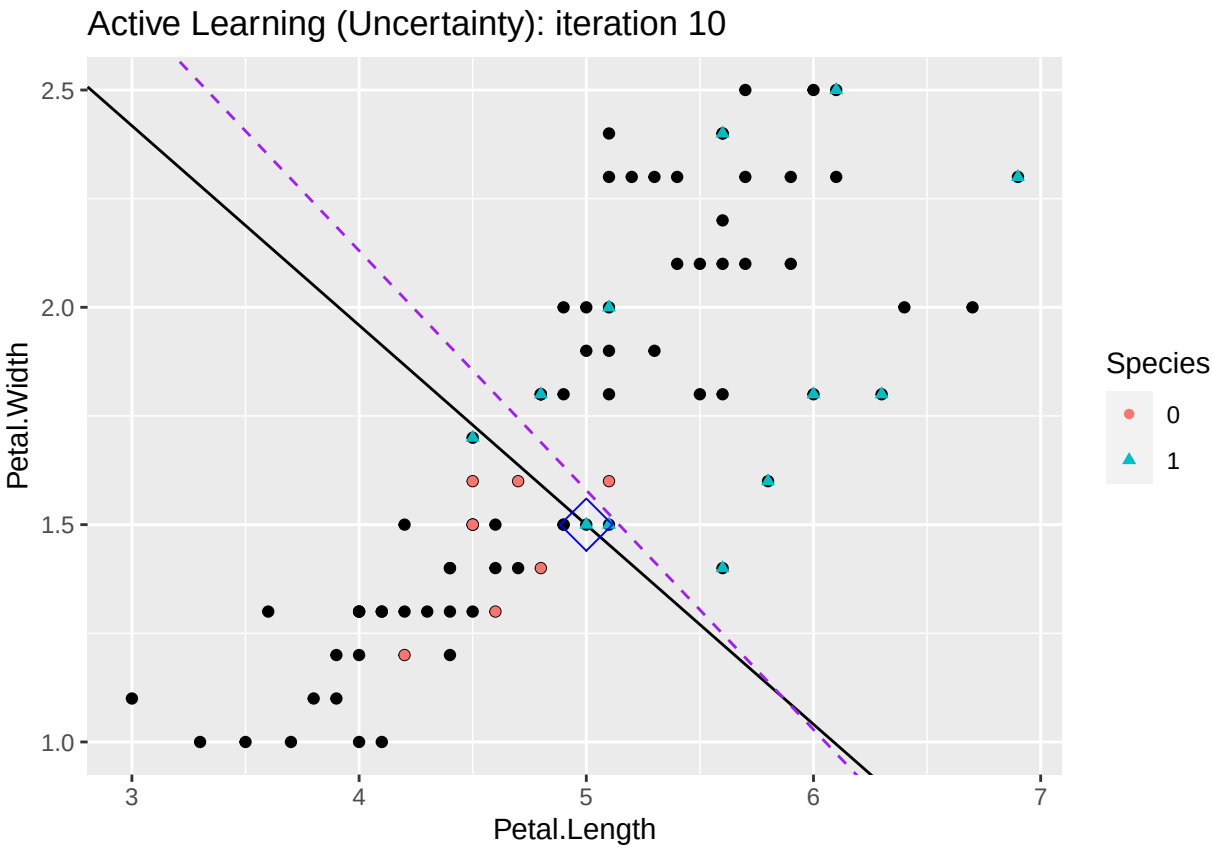
```
## [1] "done in iter 2"
```



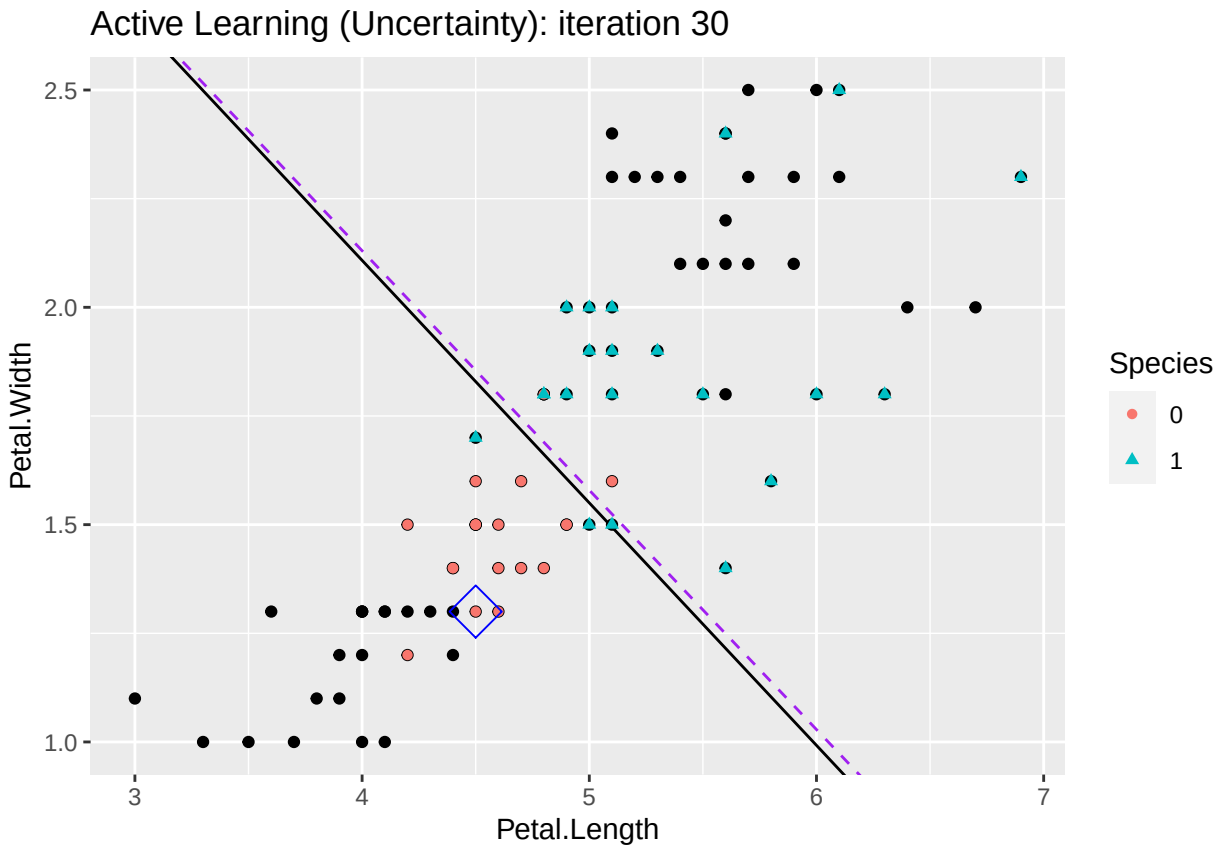

```
## [1] "done in iter 3"
## [1] "done in iter 4"
## [1] "done in iter 5"
## [1] "done in iter 6"
## [1] "done in iter 7"
## [1] "done in iter 8"
## [1] "done in iter 9"
```




```
## [1] "done in iter 10"
```



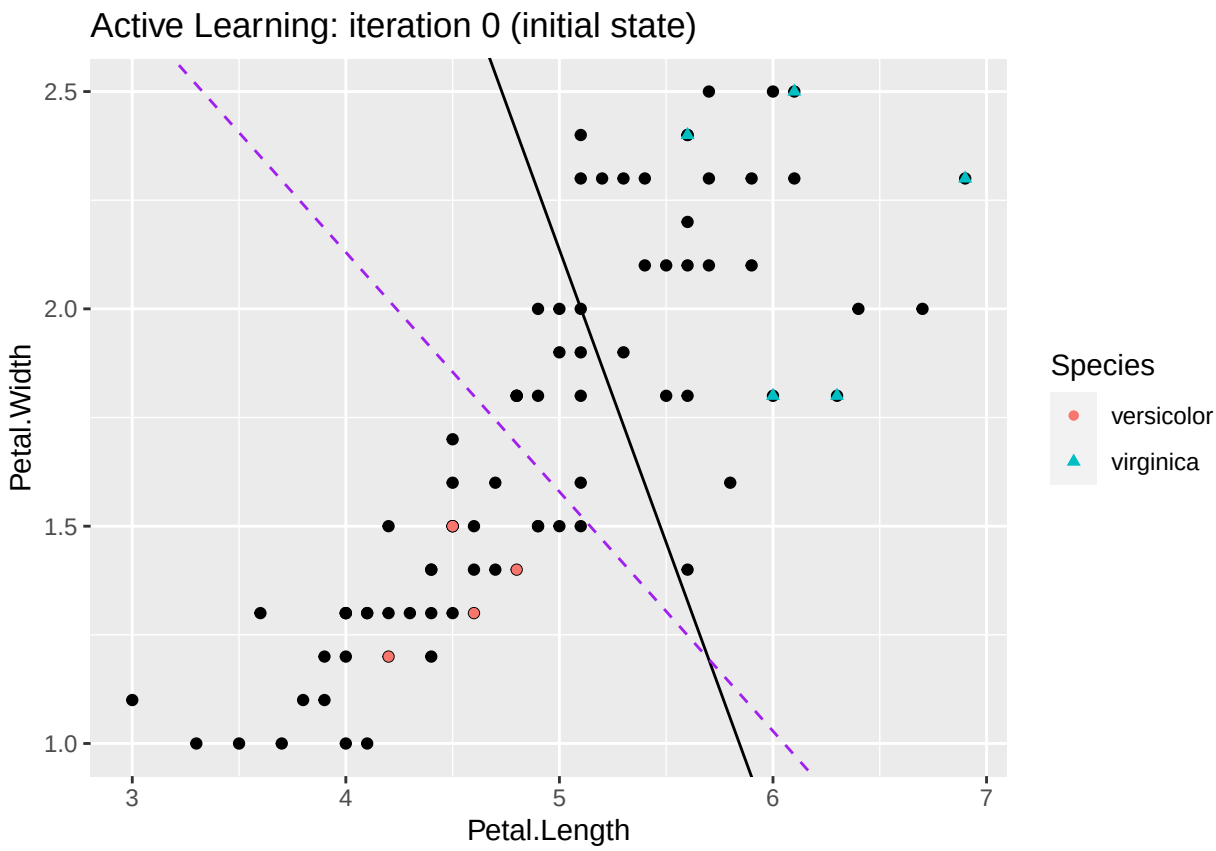

```
## [1] "done in iter 30"
```



Active Learning Querying Techniques: Random Sampling

```
iris_subset = select(iris, Petal.Length, Petal.Width, Species)
iris_subset = iris_subset[iris_subset$Species != "setosa", ]
smp_size = floor(0.8 * nrow(iris_subset))
## set the seed to make your partition reproducible
set.seed(123)
train_ind = sample(seq_len(nrow(iris_subset)), size = smp_size)

train = iris_subset[train_ind, ]
test = iris_subset[-train_ind, ]

init_ind = sample(seq_len(nrow(train)), size = 10)

train_init = train[init_ind, ]
train_pool = train[-init_ind, ]

glfit_init = glm(Species ~ ., data = train_init, family = binomial(logit))

slope_init = coef(glfit_init)[2]/(-coef(glfit_init)[3])
intercept_init = coef(glfit_init)[1]/(-coef(glfit_init)[3])

ggplot() +
geom_point(data = train, aes(x=Petal.Length, y=Petal.Width)) +
geom_abline(slope = slope_init, intercept = intercept_init) +
geom_point(data = train_init,aes(x=Petal.Length, y=Petal.Width , shape=Species, color=Species)) +
```

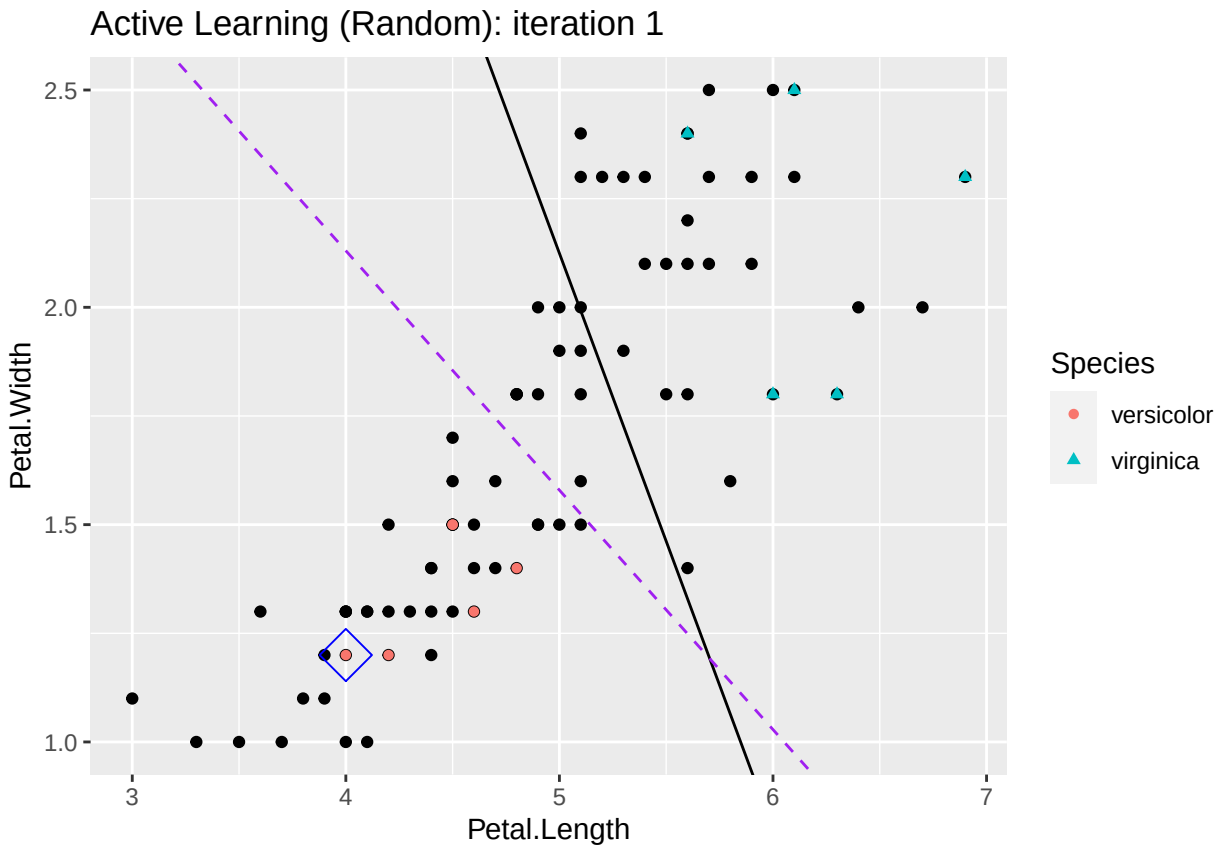
```
  ggtitle("Active Learning: iteration 0 (initial state)")+
  geom_abline(slope = truth_slope, intercept = truth_intercept, linetype = "dashed", color = "purple")
```



Select the point randomly

```
query_point_index = sample(nrow(train_pool), 1)

query_point = train_pool[query_point_index, c(1,2,3)]

train_pool = train_pool[-query_point_index,]

train_init_add = rbind(train_init, query_point)
```

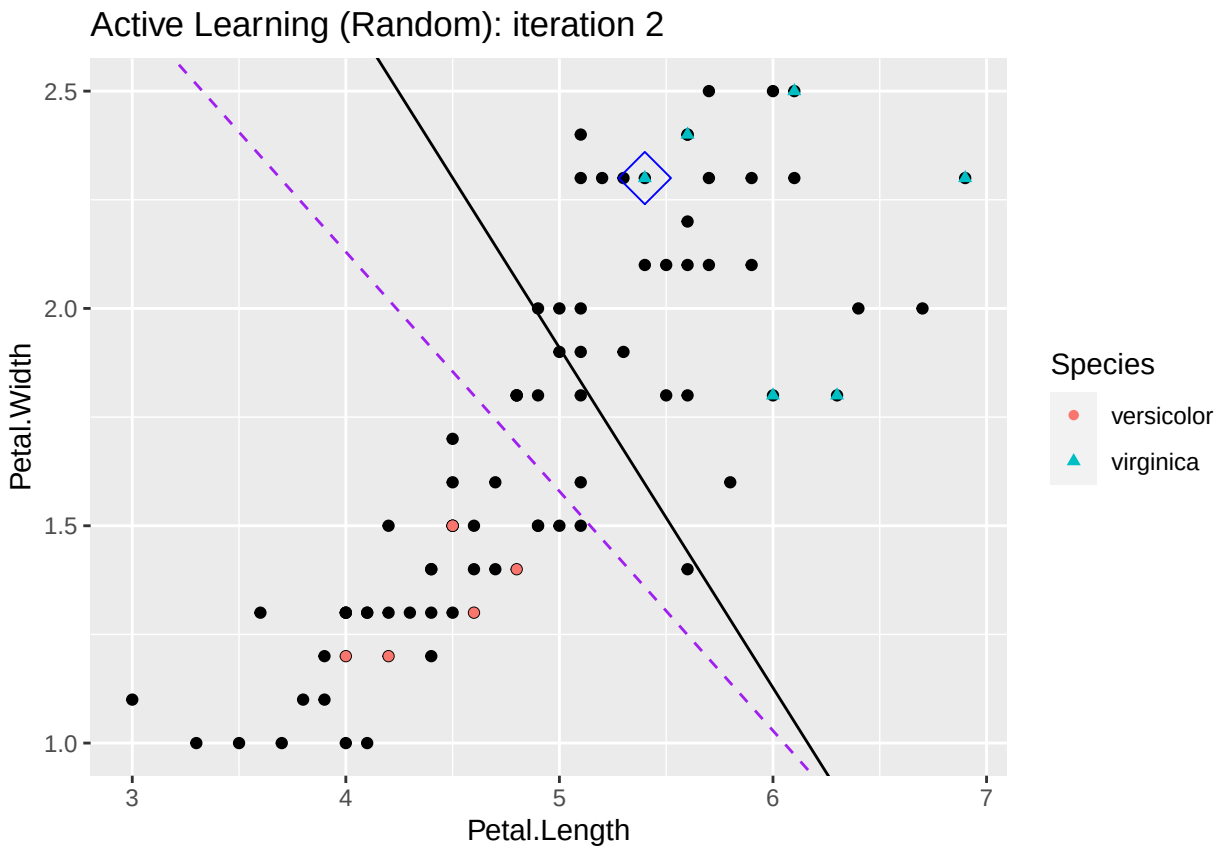
```
glfit_init_add = glm(Species ~ ., data = train_init_add, family = binomial(logit))
slope_init_add = coef(glfit_init_add)[2]/(-coef(glfit_init_add)[3])
intercept_init_add = coef(glfit_init_add)[1]/(-coef(glfit_init_add)[3])

ggplot() +
geom_point(data = train, aes(x=Petal.Length, y=Petal.Width)) +
geom_abline(slope = slope_init_add, intercept = intercept_init_add) +
geom_point(data = train_init_add,aes(x=Petal.Length, y=Petal.Width , shape=Species, color=Species)) +
geom_point(data = query_point, aes(x=Petal.Length, y=Petal.Width),
           size = 7, shape = 23, color = "blue") +
    ggtitle("Active Learning (Random): iteration 1") +
  geom_abline(slope = truth_slope, intercept = truth_intercept, linetype = "dashed", color = "purple")
```



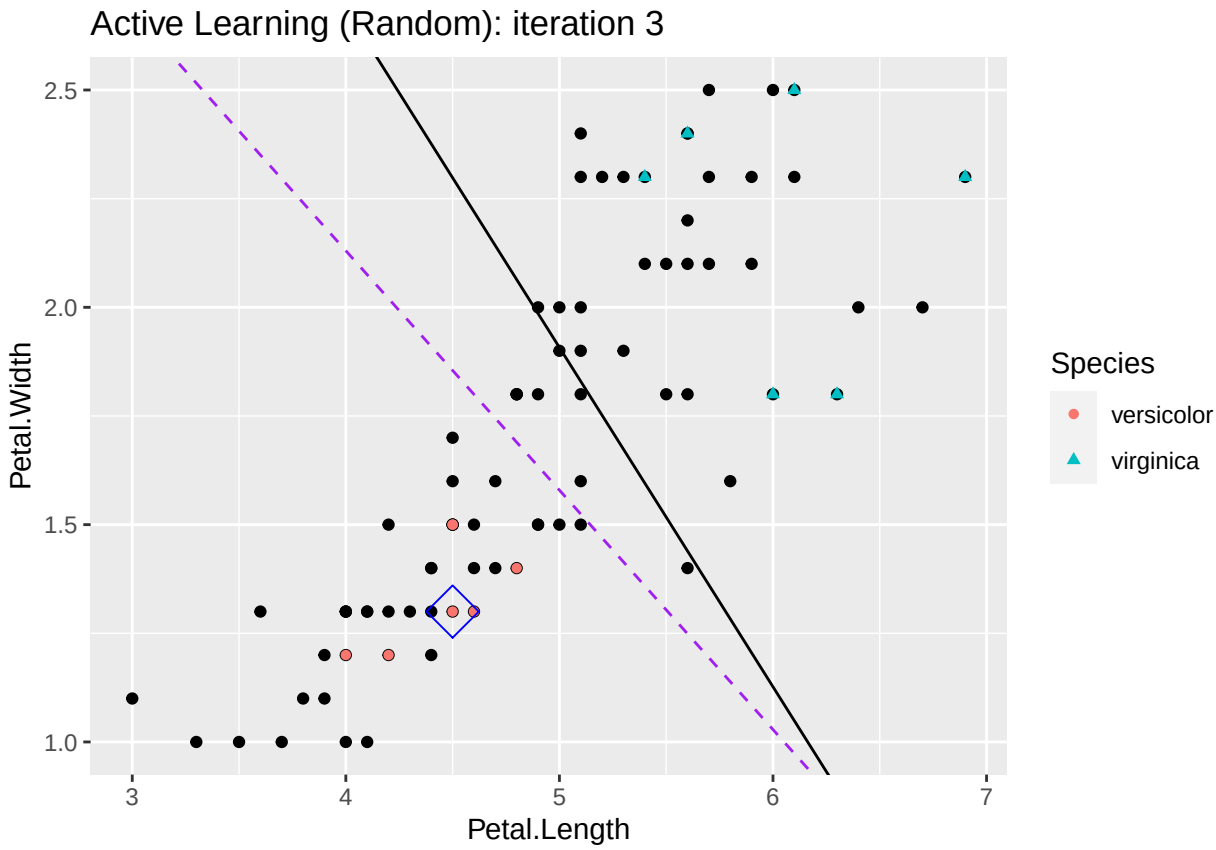

```
for (iter in 2:30){


  # query a new point
  query_point_index = sample(nrow(train_pool), 1)
  query_point = train_pool[query_point_index, c(1,2,3)]

  train_pool = train_pool[-query_point_index,]
  train_init_add = rbind(train_init_add, query_point)

  # re-train a new model
  glfit_init_add = glm(Species ~ ., data = train_init_add, family = binomial(logit))

  slope_init_add = coef(glfit_init_add)[2]/(-coef(glfit_init_add)[3])
  intercept_init_add = coef(glfit_init_add)[1]/(-coef(glfit_init_add)[3])

  curr_plot = ggplot() +
    geom_point(data = train, aes(x=Petal.Length, y=Petal.Width)) +
    geom_abline(slope = slope_init_add, intercept = intercept_init_add) +
    geom_point(data = train_init_add,aes(x=Petal.Length, y=Petal.Width , shape=Species, color=Species)) +
    geom_point(data = query_point, aes(x=Petal.Length, y=Petal.Width),
               size = 7, shape = 23, color = "blue") +
        ggtitle(paste0("Active Learning (Random): iteration ", iter))+
    geom_abline(slope = truth_slope, intercept = truth_intercept, linetype = "dashed", color = "purple")

  if(iter<=10 | iter == 30){
    ggsave(paste0("AL_process_RS_", iter, ".png"), width = 7, height = 4.33)
```

```
    print(paste0("done in iter ", iter))
  }

  if(iter%in%c(2,3,10,30)){
    print(curr_plot)
  }
}
```

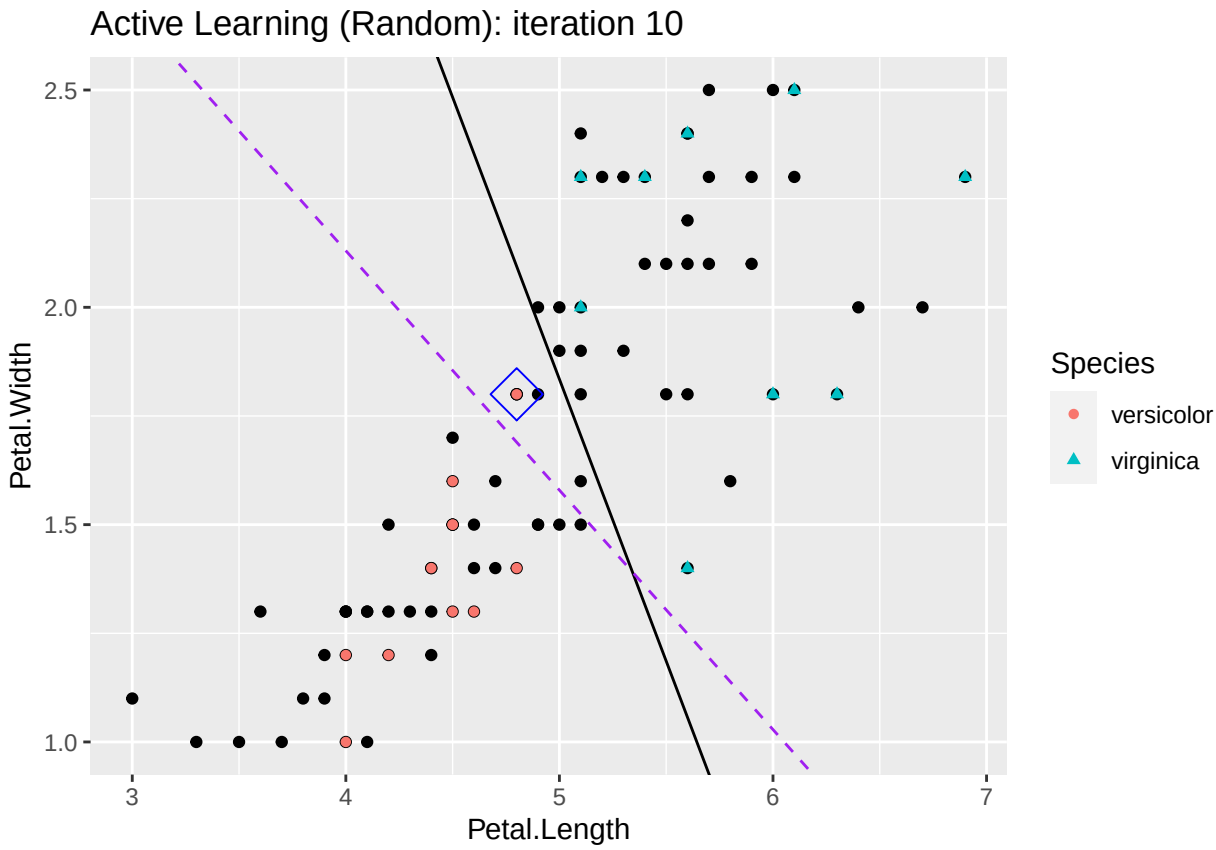
```
## [1] "done in iter 2"
```



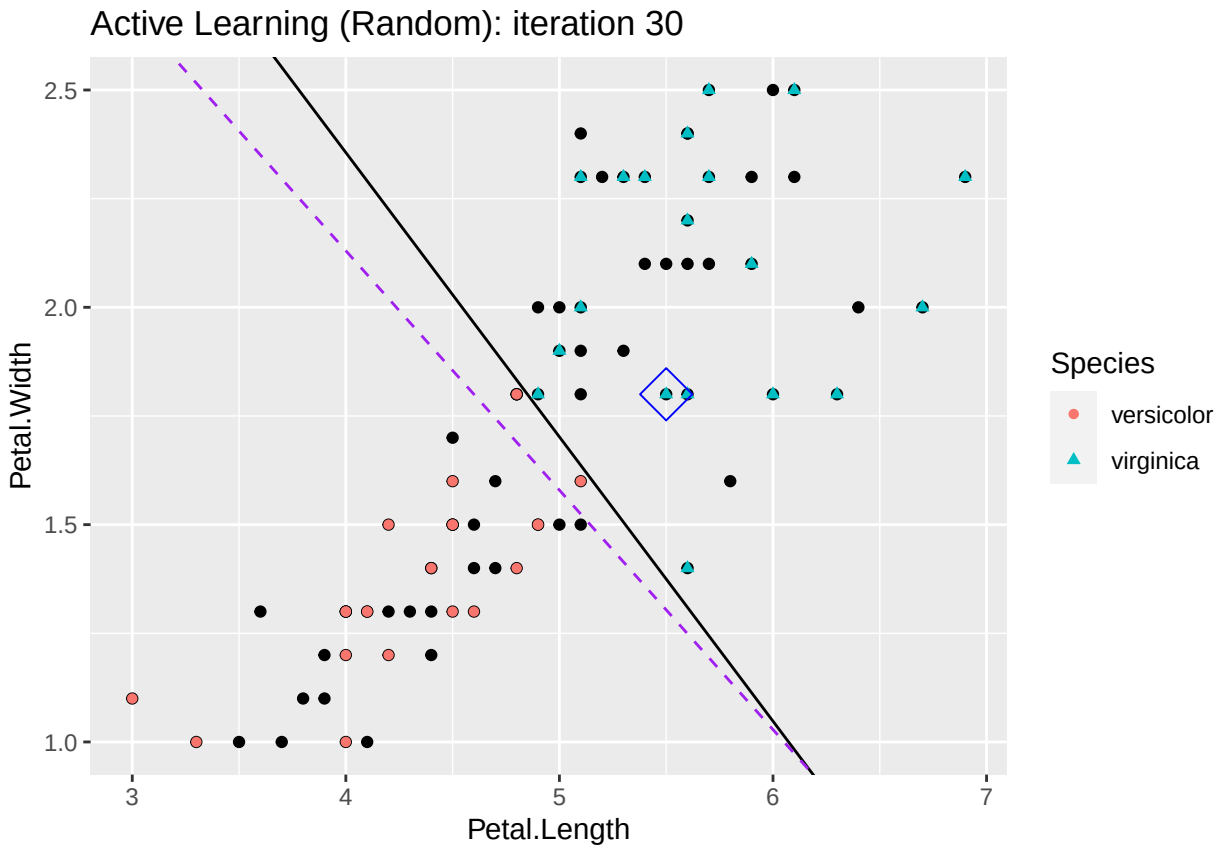

```
## [1] "done in iter 3"
## [1] "done in iter 4"
## [1] "done in iter 5"
## [1] "done in iter 6"
## [1] "done in iter 7"
## [1] "done in iter 8"
## [1] "done in iter 9"
```




```
## [1] "done in iter 10"
```




```
## [1] "done in iter 30"
```


Active Learning (Random): iteration 30
Petal.Width
Petal.Length
2.5
2.0
1.5
1.0
3
4
5
6
7
Species
versicolor
virginica