KNN Demonstration

The Part Time Economist

2024-05-09

First, we want to install the packages we will use. There are many different options that all implement KNN algos. I prefer the Tidymodels framework for machine learning in R, but it can be hard trying to explain both Tidymodels and KNN at the same time, so for this example, we will use a simpler "one liner" package for KNN that doesn't require setting up workflows or recipes.

In this example, we will use the "tidyverse" package for plotting and the "neighbr" package for predicting with KNN. If you do not have these packages installed, you will have to do that first with the install.packages("tidyverse") and install.packages("neighbr") commands

```r
library(neighbr)
library(tidyverse)
```

Creating A Dataset

Here, I am creating a fictitious dataset where the y variable is the student's outcome of failing or passing the course. The x (explanatory) variables are the hours per week the student studies and the number of classes per semester they missed.All the values are then put into a dataframe.

NOTE: creating the data via vectors and merging into a data frame is NOT needed to understand KNN, and this part can be skipped. In reality, you will typically be given a "testing" and a "training" dataset. The training dataset will have the known, labeled examples, and the testing dataset will contain the unknown data points. This portion of the code is simply included so you can recreate the dataset to follow along.

```r
#Create some simulated data
Hours_Studied <- c(1.1,1,2.3,2,3,4,5,1.1)
Classes_Skipped <- c(5,4,2.3,2,1,.5,1.3,4.6)
Course_Outcome <- c("Fail", "Fail", "Fail",
                    "Pass","Pass","Pass","Pass","Unknown")

#Arrange to Data Frame
My_Students <-
  data.frame(Course_Outcome = as.factor(Course_Outcome),
             Classes_Skipped=Classes_Skipped,
             Hours_Studied=Hours_Studied)
```

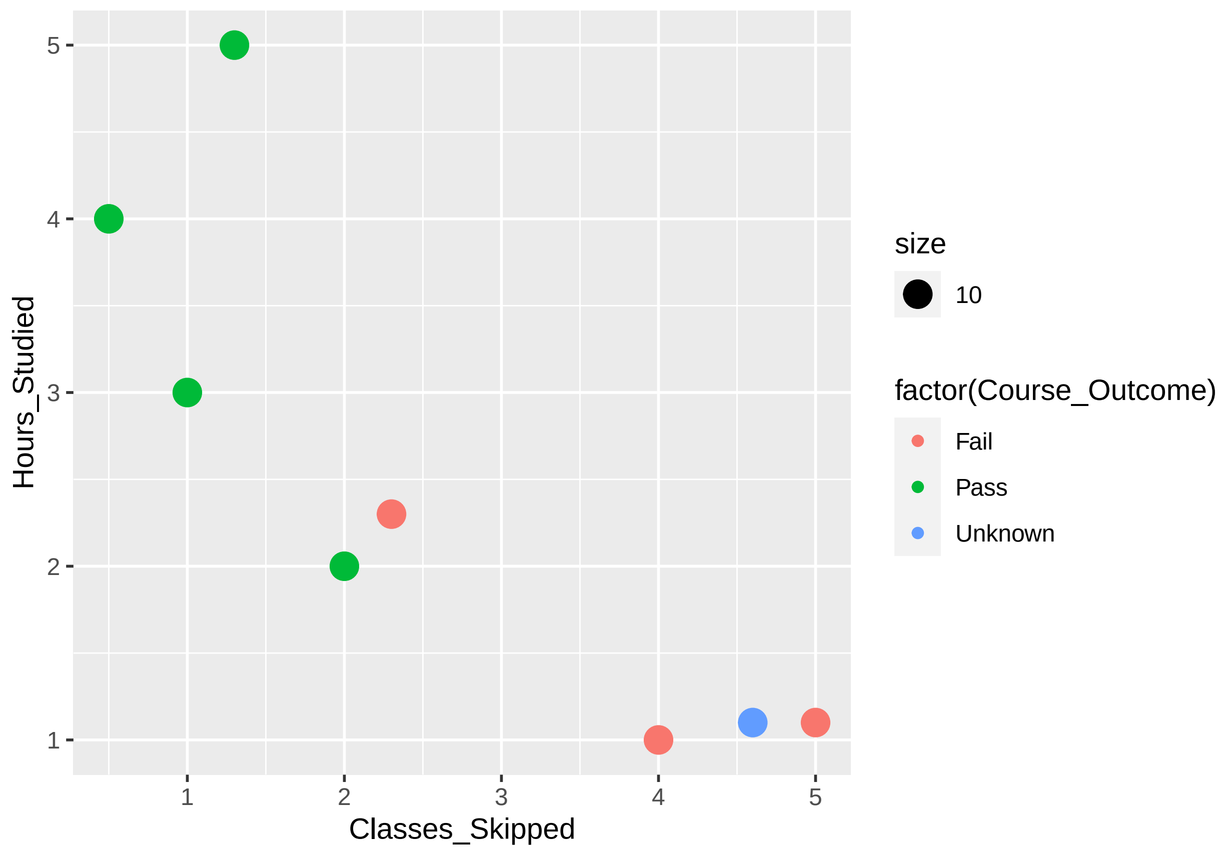
Again, this is a background step that is not necessary for you to understand. I am just creating three separate dataframes. One contains the labeled examples of known students that KNN will use for comparison. The second dataframe is the characteristics of the unknown student, and the third vector is the outcomes for the known students. If you were given a testing and training set, the only step you would need to do is to pull out the labeled examples from the dependent variable column of your training set.

```
known_examples <- My_Students[1:7,1:3]
unknown_student <-My_Students[8,2:3]
known_labels <- known_examples$Course_Outcome
```

Visualizing the Unknown Point

This code plots all the datapoints. As we can see, the unknown student is surrounded by failing studetns, so we intuitively predict their failure in the course even before running KNN.

```
ggplot(data = My_Students, mapping = aes(x = Classes_Skipped, y = Hours_Studied))+
  geom_point(aes(color = factor(Course_Outcome), size = 10))
```



Runn KNN

We then run the KNN algorithm.

```
knn_model <- knn(train_set = known_examples,
                 test_set = unknown_student,
                 categorical_target = "Course_Outcome",
                 comparison_measure="euclidean")
```

Because KNN is a “lazy learner” the prediction is done while running the model, and there is no separate predict step needed.The class predictions for our model can be found by accessing the “$categorical_target” column within the model.We can then assign the predictions to the unknown students by defining a new column and setting it to the values from the predictions.

Adding Predictions to Original Dataset

```r
prediction <- knn_model$test_set_scores$categorical_target

unknown_student$Prediction <- prediction
```

Note: Because KNN is a distance-based algo, we will generally want to scale our data before running the model. Some KNN packages do this automatically, and some do not. If you need to scale your data, it can be done using the Tidymodels "step" functions or manually using the base R "scale" command.

```r
scaled_data_frame <- scale(My_Students[1:8,2:3])
```