EDX Data Science Project HarvardX

Miami Kelvin

2019-06-17

Introduction to machine learning

The Data Science course on EDX by Harvard has introduced me to some core data science concepts, i am certainly not yet a seasoned data scientist but with the little knowledge i have received from the course i will attempt to demonstrate machine learning through this ML capstone project. This ML projects offer you a a slightly naive approach to training and testing a Machine Learning algorith to categorise some images.

Problem Statement

At my current job, we recently engaged a suplier to digitize all our client documents. one of the requirements was to categorise documents based on type and unit and at the same time embed metadata elements. Their approach was to have human workers look through each and every document and categorize it, the project was for 140,000 files and on average each file has 30 different document types varying in content and structure which can be 1 of multiple types(approximately 42 not counting emails and unstructured documents). I saw this as a total waste of time and resources, this is something that could be done more efficiently using a well trained algorithem. While I am miles away from this, this will be a great way to learn. Bingo! We have a great problem to implement a classification algorithm, this would have been a brillian project but taking a few steps back i realized i could not proceed with that given the timelines and the project requirements that data is available for peer review, so i went for the next BIG thing, Iris. This project will attempt a very naive classification algorithm.

Dataset overview

There are very many datasets out in the wild however for a learning algorithm for image recognition and simple enough for a learner new to R and machine learning, the Iris data set is perfect. Description and download of the iris dataset can be found at https://archive.ics.uci.edu/ml/datasets/Iris

Getting the data

We are going to download the data and save it in our working directory inside data directory. We could simply do data(iris) and be done with that, but a real project would involve some data imports of sort. So we emulate this, but first some house keeping ## do some housekeeping

```
# get curent working directory
cwd <- getwd()
dataSourceUrl <- "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

print(paste("the curent working directory is ",cwd))
```

```
## [1] "the curent working directory is  D:/projects/data_science_ML_capstone"
```

```
# check if the data directory exists
datadir <- "data"
if(!dir.exists(file.path(cwd, datadir))){
  print("data direcory does not exist it will be created")
  #a try catch statement can help raise a warning if there are permission issues
  dir.create(file.path(cwd, datadir))
}else{
```

```
  print("directory structure is ok we will now download iris dataset")
}
```

```
## [1] "directory structure is ok we will now download iris dataset"
```

```
datapath <- file.path(cwd, datadir)
dataFilePath <- file.path(datapath, 'iris.data')
dataFilePath
```

```
## [1] "D:/projects/data_science_ML_capstone/data/iris.data"
```

```
if(file.exists(dataFilePath)){
  print("data file exists")
} else {
  print("data file does not exist, we will download it to the data directory")
  download.file(dataSourceUrl, dataFilePath)
}
```

```
## [1] "data file exists"
```

```
# Load iris dataset

#conventional knowledge tells us this is a csv that does not have headers but we will preview it all th
irisCsv <- read_csv(dataFilePath, col_names = c('sepal_length_cm','sepal_width_cm','petal_length_cm','p

head(irisCsv, 6)
```

```
## # A tibble: 6 x 5
##   sepal_length_cm sepal_width_cm petal_length_cm petal_width_cm species
##             <dbl>          <dbl>           <dbl>          <dbl> <fct>
## 1             5.1            3.5             1.4            0.2 Iris-setosa
## 2             4.9            3               1.4            0.2 Iris-setosa
## 3             4.7            3.2             1.3            0.2 Iris-setosa
## 4             4.6            3.1             1.5            0.2 Iris-setosa
## 5             5              3.6             1.4            0.2 Iris-setosa
## 6             5.4            3.9             1.7            0.4 Iris-setosa
```

```
# data was imported as list, we would like it as tible or dataframe
# we can assign collumn names here as well
irisDf <- data.frame(irisCsv)
names(irisDf) <- c('sepal_length_cm','sepal_width_cm','petal_length_cm','petal_width_cm', 'species')
```

Performing data cleaning and wrangling

this dataset is a relatively clean we do not need to do cleaning we will replace all occurance of Iris before the species name instean

```
irisDf <- irisDf %>% mutate(species = str_replace(species, 'Iris-', ''))
```

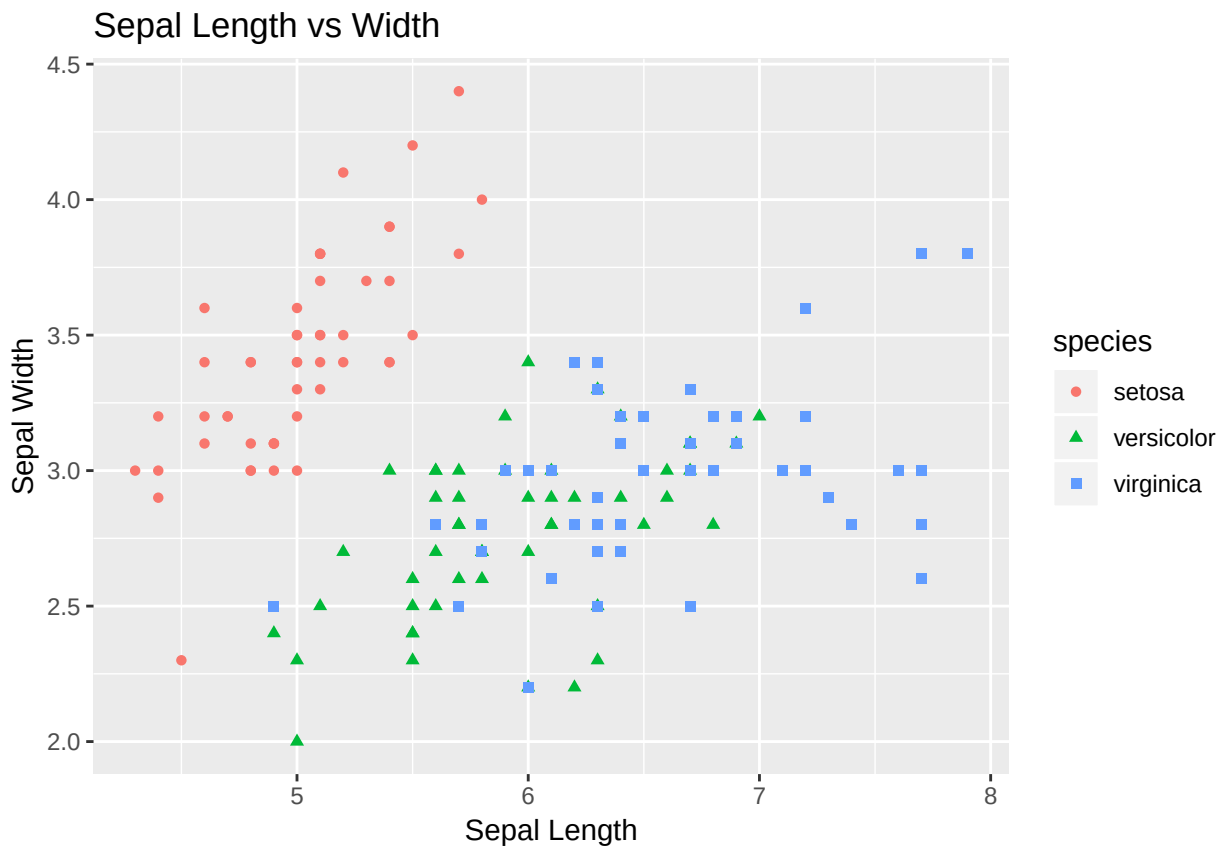
Summarise the dataset we will use

This will give us a basis for our assumption in the classification

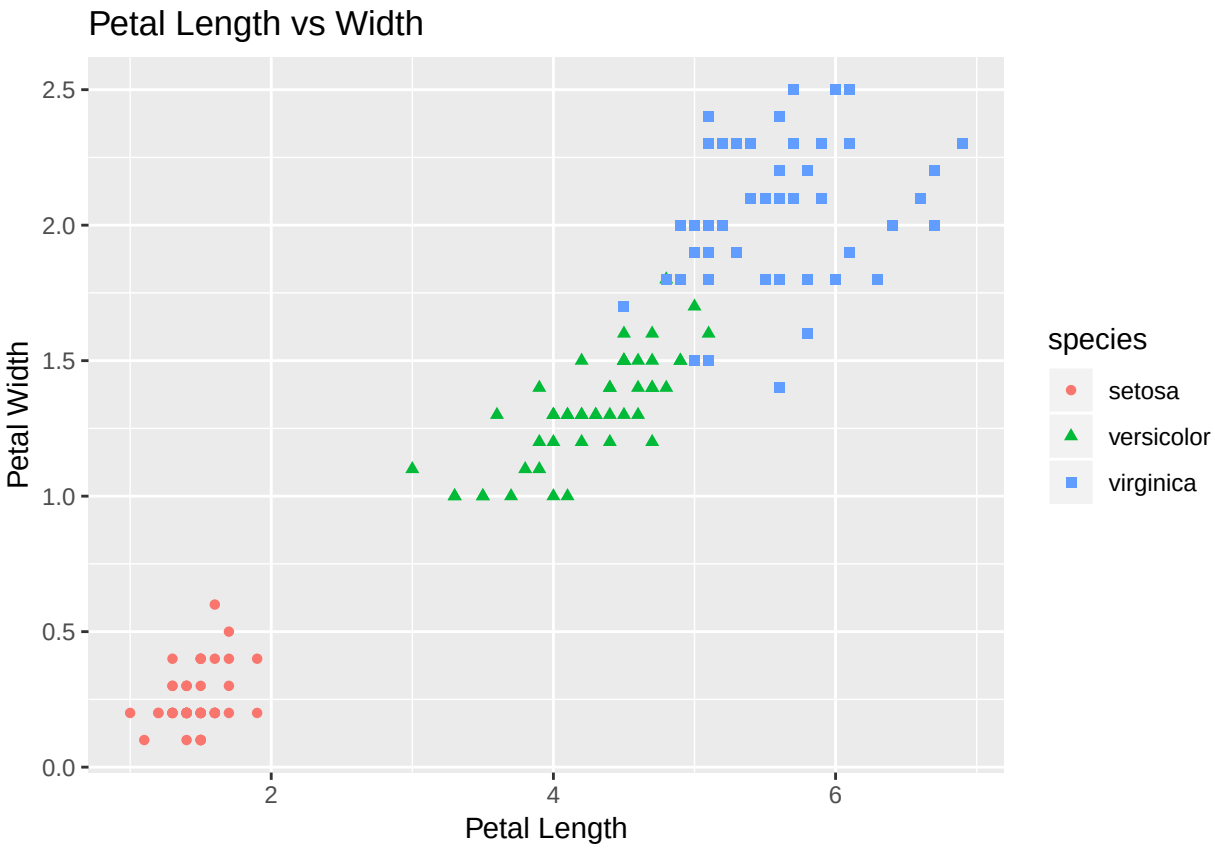
```
summary <- irisDf %>% dplyr::group_by(species) %>% dplyr::summarise(    avg_sepl = mean(sepal_length_cm
    avg_petl = mean(petal_length_cm), sd_petl = sd(petal_length_cm),
    avg_petw = mean(petal_width_cm), sd_petw = sd(petal_width_cm))
summary
```

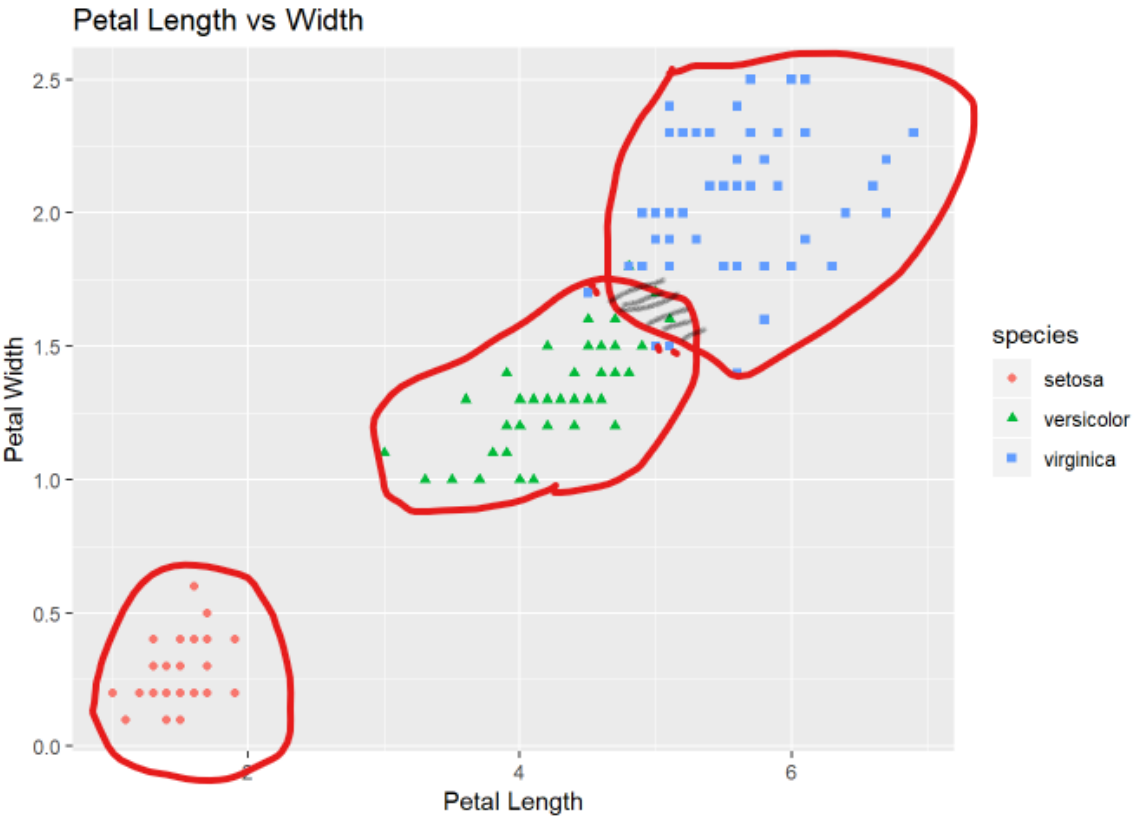
```
## # A tibble: 3 x 9
##   species avg_sepl sd_sepl avg_sepw sd_sepw avg_petl sd_petl avg_petw
##   <chr>      <dbl>   <dbl>    <dbl>   <dbl>    <dbl>   <dbl>    <dbl>
## 1 setosa      5.01   0.352     3.42   0.381     1.46   0.174    0.244
## 2 versic~     5.94   0.516     2.77   0.314     4.26   0.470    1.33
## 3 virgin~     6.59   0.636     2.97   0.322     5.55   0.552    2.03
## # ... with 1 more variable: sd_petw <dbl>
```

```
# Now we check the Sepal Length vs Width. Notice the sepal dimensions for the Virginica and Versicolor
irisDf %>% ggplot(aes(x = sepal_length_cm, y = sepal_width_cm)) +
  geom_point(aes(color=species, shape=species)) +
  xlab("Sepal Length") +
  ylab("Sepal Width") +
  ggtitle("Sepal Length vs Width")
```




```
irisDf %>% ggplot(aes(x = petal_length_cm, y = petal_width_cm)) +
  geom_point(aes(color=species, shape=species)) +
  xlab("Petal Length") +
  ylab("Petal Width") +
  ggtitle("Petal Length vs Width")
```


Petal Length vs Width
Petal Width
Petal Length
2.5
2.0
1.5
1.0
0.5
0.0
2
4
6
species
setosa
versicolor
virginica


Petal Length vs Width
Petal Width
Petal Length
2.5
2.0
1.5
1.0
0.5
0.0
4
6
species
setosa
versicolor
virginica

With this simple scatter plot some patterns/features begin to emerge in covarriance of the two variables of the petal feature, we can use as a basis for our classification

Creating a model

Split the data to Training and Test sets

```
#define our observations
y <- irisDf$species
set.seed(2)

test_index <- createDataPartition(y,times=1,p=0.4,list=FALSE)

test <- irisDf[test_index,]

train <- irisDf[-test_index,]
```

y_hat <- sample(c("setosa","versicolor", "virginica"), length(test_index), replace=TRUE) %>% factor(levels = levels(test$Species))

Training the Model

We will train a simple model using the discoveries we have made so far

```
petl_cutofs <- seq(min(train$petal_length_cm), max(train$petal_length_cm), 0.1)
petw_cutofs <- seq(min(train$petal_width_cm), max(train$petal_width_cm), 0.1)

print(petw_cutofs)
```

```
##  [1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7
## [18] 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5
```

```
petl_accu <- map_dbl(petl_cutofs, function(x){
    y_hat <- ifelse(train$petal_length_cm < x, "setosa", {ifelse(train$petal_length_cm > x, "virginica"
    #%>% factor(levels = levels(test$species))
    mean(y_hat == train$species)
})
mean(petl_accu)
```

```
## [1] 0.5621469
```

```
petw_accu <- map_dbl(petw_cutofs, function(x){
    y_hat <- ifelse(train$petal_width_cm < x, "setosa", { ifelse( train$petal_width_cm > x , "virginica
    #%>% factor(levels = levels(test$species))
    mean(y_hat == train$species)
})
print(petw_accu)
```

```
##  [1] 0.3333333 0.3777778 0.6000000 0.6000000 0.6555556 0.6666667 0.6666667
##  [8] 0.6666667 0.6666667 0.7222222 0.6777778 0.6666667 0.6666667 0.6666667
## [15] 0.6666667 0.6666667 0.6444444 0.5666667 0.5222222 0.4777778 0.4444444
## [22] 0.4222222 0.3666667 0.3444444 0.3333333
```

```
print(petl_accu)
```

```
##  [1] 0.3333333 0.3444444 0.3444444 0.4777778 0.4777778 0.5777778 0.6555556
```

```
##  [8] 0.6555556 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667
## [15] 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667
## [22] 0.6666667 0.6666667 0.6666667 0.6666667 0.6777778 0.6777778 0.6666667
## [29] 0.6666667 0.7111111 0.7000000 0.6888889 0.6666667 0.6888889 0.6888889
## [36] 0.6777778 0.6888889 0.6444444 0.6444444 0.6111111 0.5666667 0.5444444
## [43] 0.5333333 0.5222222 0.4888889 0.4555556 0.4222222 0.3888889 0.3777778
## [50] 0.3777778 0.3666667 0.3666667 0.3666667 0.3555556 0.3555556 0.3555556
## [57] 0.3444444 0.3444444 0.3333333
```

```
best_cutof_w <- petw_cutofs[which.max(petw_accu)]
best_cutof_l <- petl_cutofs[which.max(petl_accu)]
#we mutate the df and add the coefficient of variation that we can use in our classifier

summary %>% mutate(cv_petl= sd_petl/avg_petl,cv_petw= sd_petw/avg_petw) %>% select(cv_petw, cv_petl)
```

```
## # A tibble: 3 x 2
##   cv_petw cv_petl
##     <dbl>   <dbl>
## 1   0.439  0.119
## 2   0.149  0.110
## 3   0.136  0.0994
```

Best cutoff

Our best cutoff are

```
petl_cutofs <- seq(min(train$petal_length_cm), max(train$petal_length_cm), 0.1)
petw_cutofs <- seq(min(train$petal_width_cm), max(train$petal_width_cm), 0.1)

print(petw_cutofs)
```

```
##  [1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7
## [18] 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5
```

```
petl_accu <- map_dbl(petl_cutofs, function(x){
    y_hat <- ifelse(train$petal_length_cm < x, "setosa", {ifelse(train$petal_length_cm > x, "virginica"
    #%>% factor(levels = levels(test$species))
    mean(y_hat == train$species)
})
mean(petl_accu)
```

```
## [1] 0.5621469
```

```
petw_accu <- map_dbl(petw_cutofs, function(x){
    y_hat <- ifelse(train$petal_width_cm < x, "setosa", { ifelse( train$petal_width_cm > x , "virginica
    #%>% factor(levels = levels(test$species))
    mean(y_hat == train$species)
})
print(petw_accu)
```

```
##  [1] 0.3333333 0.3777778 0.6000000 0.6000000 0.6555556 0.6666667 0.6666667
##  [8] 0.6666667 0.6666667 0.7222222 0.6777778 0.6666667 0.6666667 0.6666667
## [15] 0.6666667 0.6666667 0.6444444 0.5666667 0.5222222 0.4777778 0.4444444
## [22] 0.4222222 0.3666667 0.3444444 0.3333333
```

```
print(petl_accu)
```

```
##  [1] 0.3333333 0.3444444 0.3444444 0.4777778 0.4777778 0.5777778 0.6555556
```

```
##  [8] 0.6555556 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667
## [15] 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667 0.6666667
## [22] 0.6666667 0.6666667 0.6666667 0.6666667 0.6777778 0.6777778 0.6666667
## [29] 0.6666667 0.7111111 0.7000000 0.6888889 0.6666667 0.6888889 0.6888889
## [36] 0.6777778 0.6888889 0.6444444 0.6444444 0.6111111 0.5666667 0.5444444
## [43] 0.5333333 0.5222222 0.4888889 0.4555556 0.4222222 0.3888889 0.3777778
## [50] 0.3777778 0.3666667 0.3666667 0.3666667 0.3555556 0.3555556 0.3555556
## [57] 0.3444444 0.3444444 0.3333333
```

```
best_cutof_w <- petw_cutofs[which.max(petw_accu)]
best_cutof_l <- petl_cutofs[which.max(petl_accu)]
#we mutate the df and add the coefficient of variation that we can use in our classifier

summary %>% mutate(cv_petl= sd_petl/avg_petl,cv_petw= sd_petw/avg_petw) %>% select(cv_petw, cv_petl)
```

```
## # A tibble: 3 x 2
##   cv_petw cv_petl
##     <dbl>   <dbl>
## 1   0.439  0.119
## 2   0.149  0.110
## 3   0.136  0.0994
```

Testing themodel on the Test Set

Evaluating accuracy of the test set

Including Plots