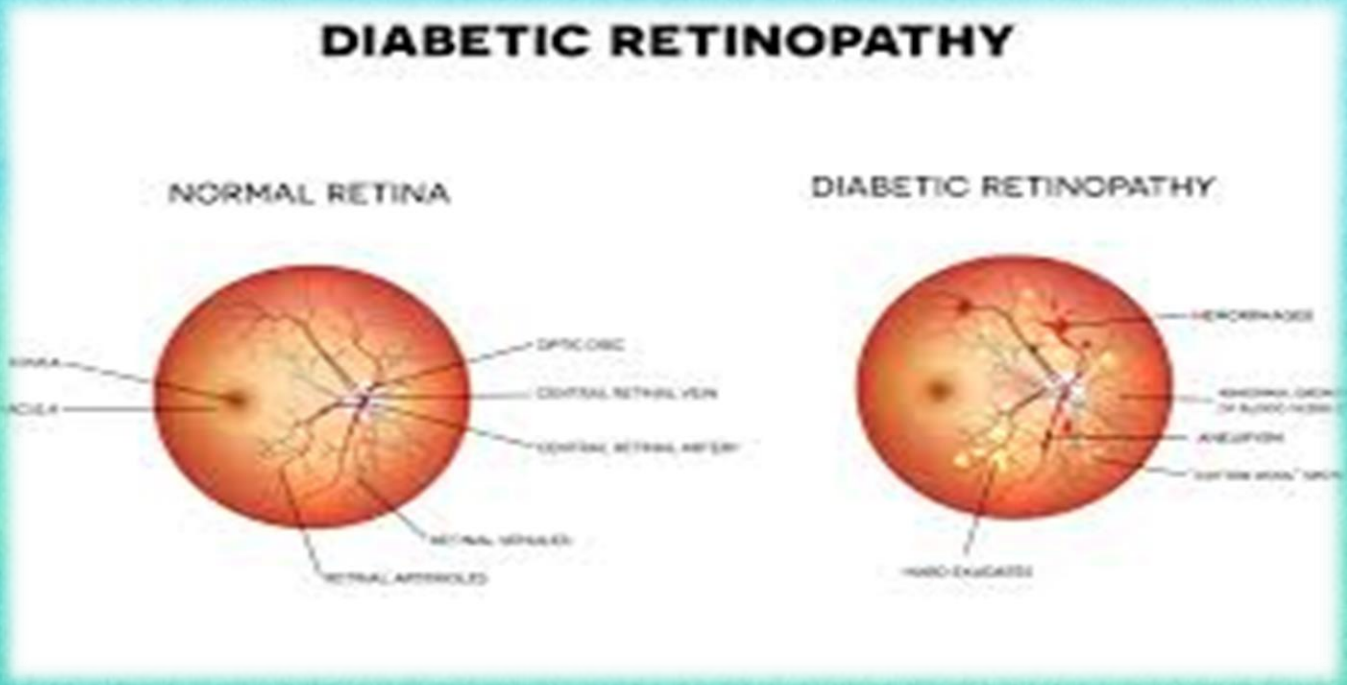


Diabetic retinopathy

Subject: Advances in data sciences
Prof: Chaiyaporn Mutsalklisana
Team: Data Geeks

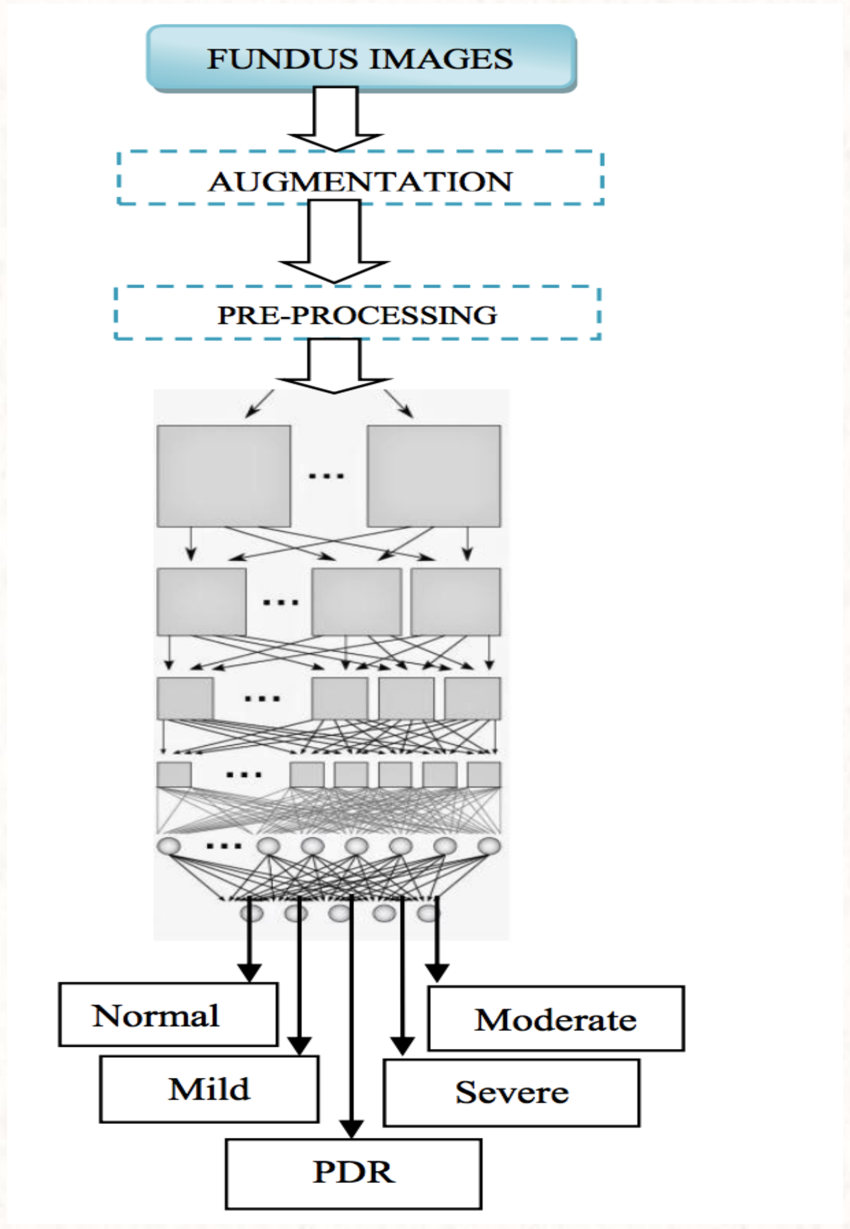
DIABETIC RETINOPATHY

- Diabetic Retinopathy is also known as Diabetic Eye Disease, is when damage occurs to the retina due to diabetes.
- Experts have categorized diabetic diabetic retinopathy into five stages:
  - Normal
  - Mild
  - Severe
  - Nonproliferative(NPDR)

- Our proposed methodology strongly emerged based on the key aspects of diseases severity classification from the fundus images.
- The basic steps that can be used to achieve maximum accuracy are
  - Data Augmentation
  - Pre-processing
  - Initialization of networks
  - Training
  - Activation Function Selections
  - Regularizations
  - Ensemble the multiple methods

Building Blocks of our architecture

- Data Augmentation
- Preprocessing
- Deep Convolutional Neural Network
- Classification



Past Diagnosis on Diabetic Retinopathy

- Visual Acuity test: This uses a eye chart to measure how well a person can see at various distances.

- Pupil dilation: The eye care professional places drops into the eye to dilate the pupil.

- *Ophthalmoscopy* or *fundus photography*:Ophthalmoscopy is an examination of the retina in which the eye care professional: (1) looks through a slit lamp biomicroscope with a special magnifying lens that provides a narrow view of the retina, or (2) wearing a headset (indirect ophthalmoscope) with a bright light, looks through a special magnifying glass and gains a wide view of the retina. Hand-held ophthalmoscopy is insufficient to rule out significant and treatable diabetic retinopathy.

- *Fundus Fluorescein angiography (FFA)*: This is an imaging technique which relies on the circulation of Fluorescein dye to show staining, leakage, or non-perfusion of the retinal and choroidal vasculature.

- *Optical coherence tomography (OCT)*:This is an optical imaging modality based upon interference, and analogous to ultrasound. It produces cross-sectional images of the retina (B-scans) which can be used to measure the thickness of the retina and to resolve its major layers, allowing the observation of swelling.

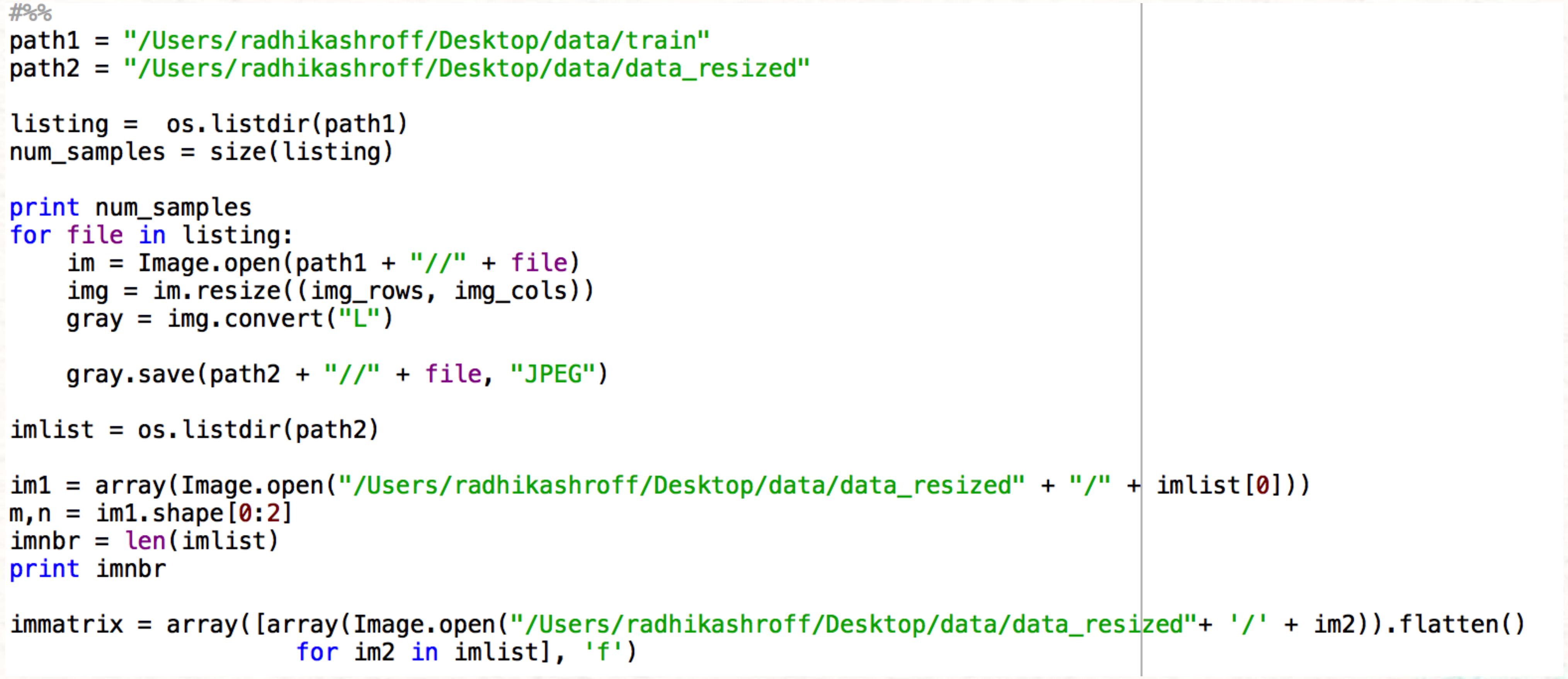
The eye care professional will look at the retina for early signs of the disease, such as:

- leaking blood vessels,
- retinal swelling, such as macular edema,
- pale, fatty deposits on the retina (exudates) – signs of leaking blood vessels,
- damaged nerve tissue (neuropathy), and
- any changes in the blood vessels.

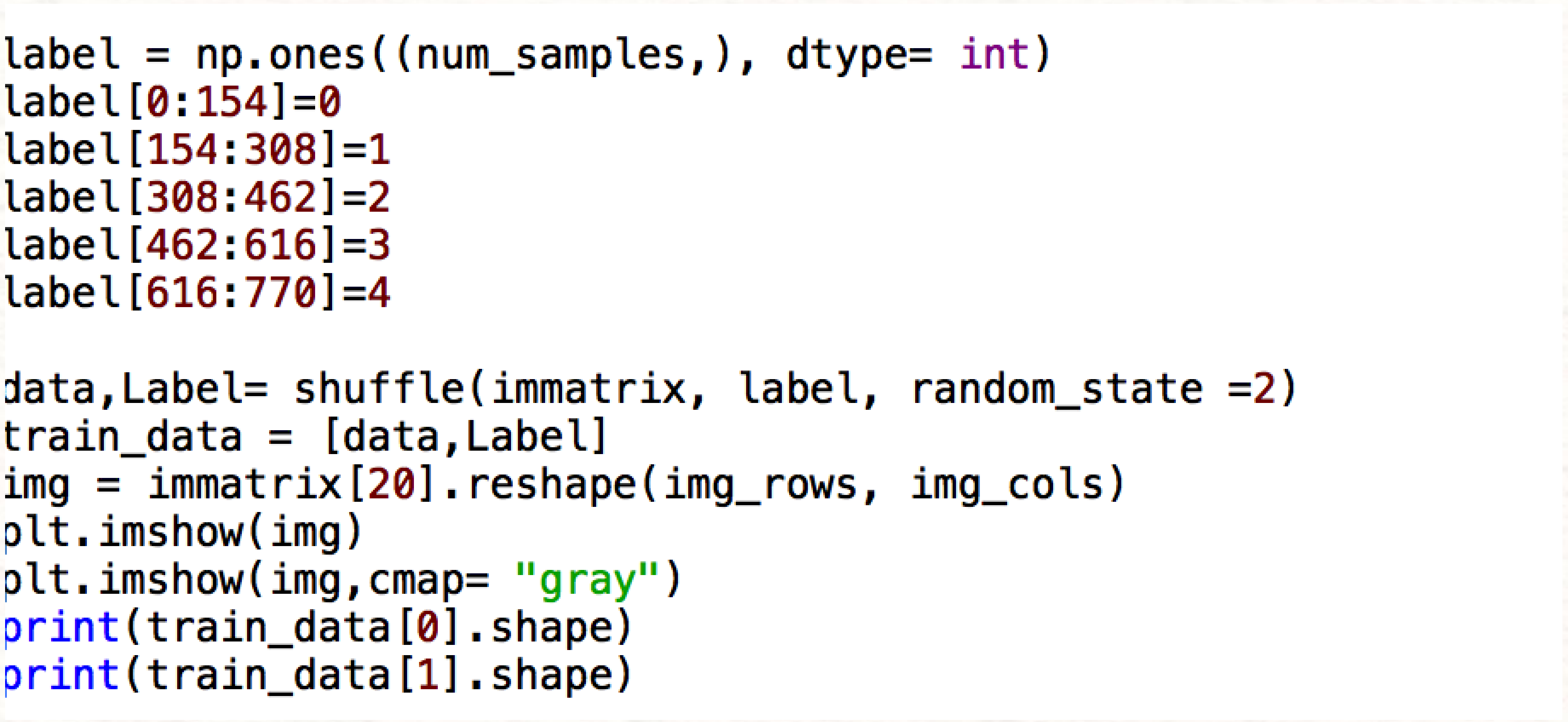
Convolutional Neural Networks

- CNNs are advanced Artificial Neural Networks in which connectivity pattern between the neurons is localized.
- CNN implementation is defined by the following steps
- Convolution
- Subsampling
- Activation
- Fully connected
- Loss

```python
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, RMSprop, adam
from keras.utils import np_utils

import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import os
import theano

from PIL import Image

from numpy import *
from sklearn.utils import shuffle
from sklearn.cross_validation import train_test_split
```

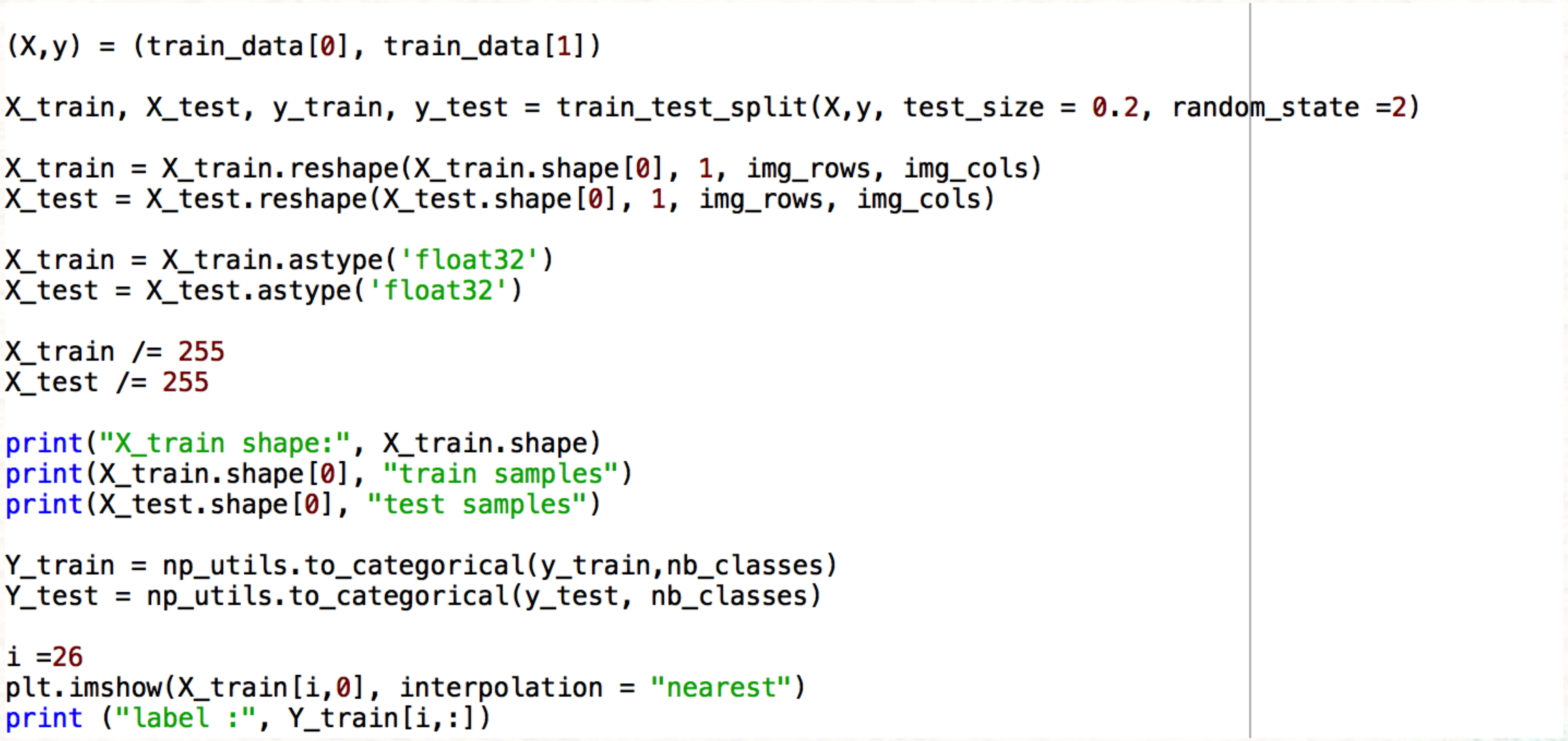


```
#%%
path1 = "/Users/radhikashroff/Desktop/data/train"
path2 = "/Users/radhikashroff/Desktop/data/data_resized"

listing =  os.listdir(path1)
num_samples = size(listing)

print num_samples
for file in listing:
    im = Image.open(path1 + "//" + file)
    img = im.resize((img_rows, img_cols))
    gray = img.convert("L")

    gray.save(path2 + "//" + file, "JPEG")

imlist = os.listdir(path2)

im1 = array(Image.open("/Users/radhikashroff/Desktop/data/data_resized" + "/" + imlist[0]))
m,n = im1.shape[0:2]
imnbr = len(imlist)
print imnbr

immatrix = array([array(Image.open("/Users/radhikashroff/Desktop/data/data_resized"+ '/' + im2)).flatten()
                   for im2 in imlist], 'f')
```

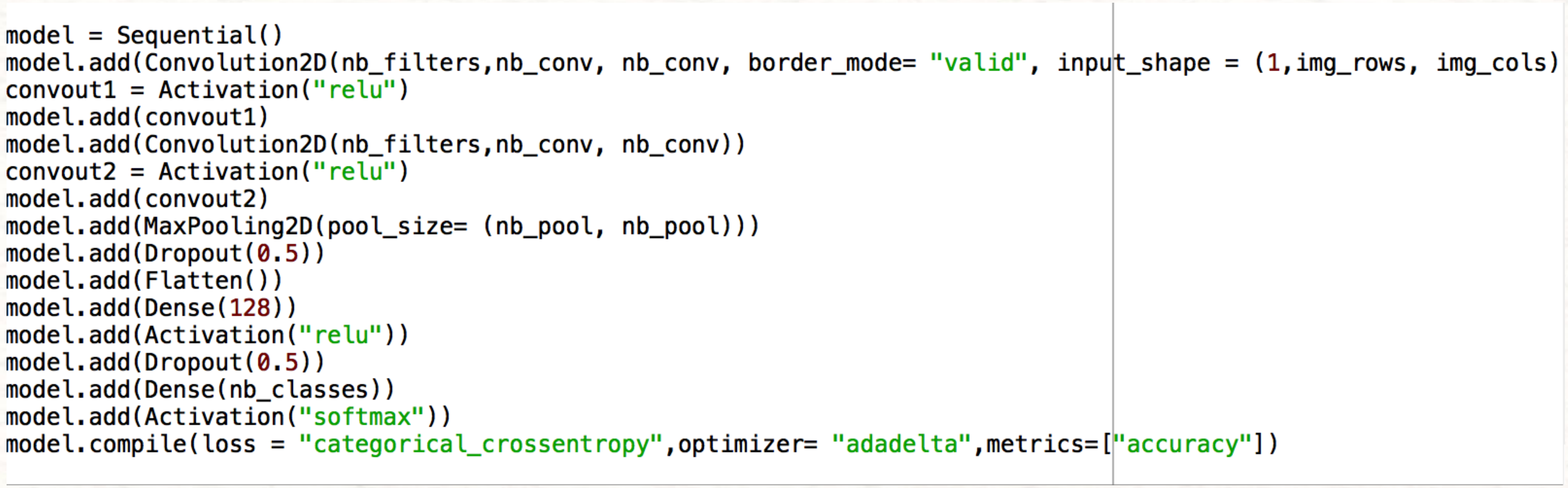


```python
label = np.ones((num_samples,), dtype= int)
label[0:154]=0
label[154:308]=1
label[308:462]=2
label[462:616]=3
label[616:770]=4

data,Label= shuffle(immatrix, label, random_state =2)
train_data = [data,Label]
img = immatrix[20].reshape(img_rows, img_cols)
plt.imshow(img)
plt.imshow(img,cmap= "gray")
print(train_data[0].shape)
print(train_data[1].shape)
```

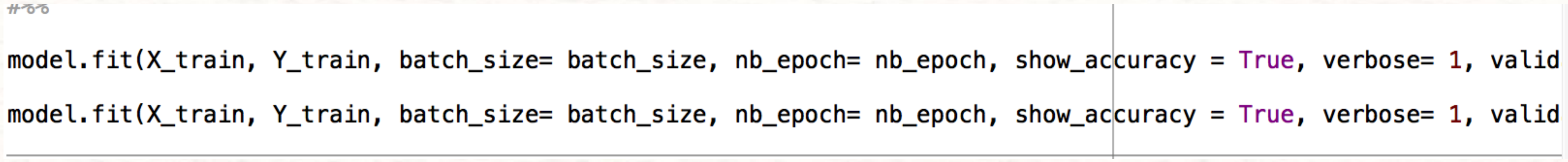
```
#%%
batch_size = 50
nb_classes = 5
nb_epoch = 50

img_rows, img_cols = 200,200

img_channels =1

nb_filters = 32

nb_pool =2

nb_conv =3
```



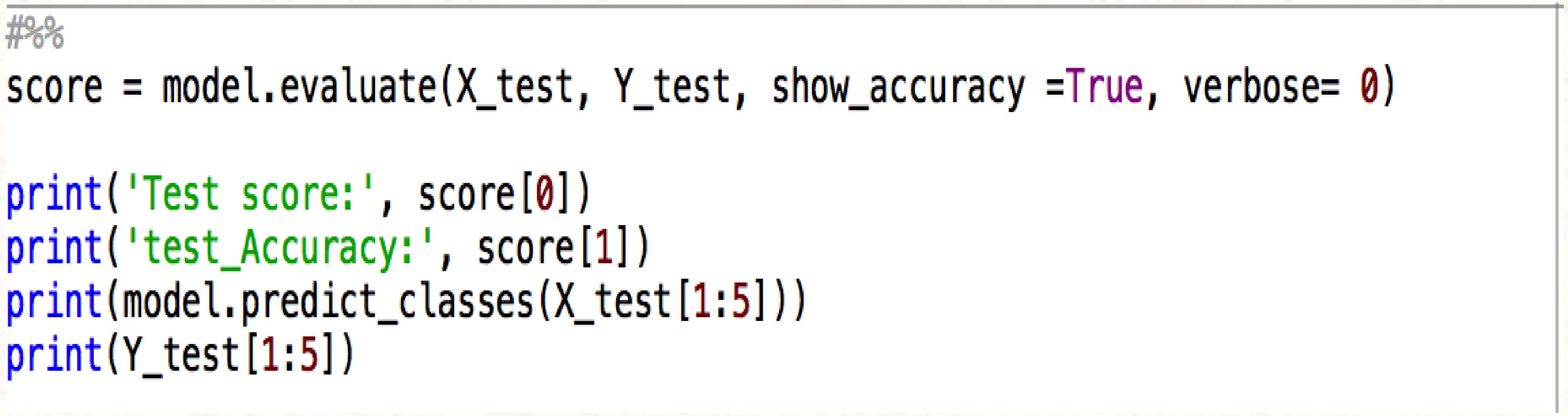
```
(X,y) = (train_data[0], train_data[1])

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state =2)

X_train = X_train.reshape(X_train.shape[0], 1, img_rows, img_cols)
X_test = X_test.reshape(X_test.shape[0], 1, img_rows, img_cols)

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

X_train /= 255
X_test /= 255

print("X_train shape:", X_train.shape)
print(X_train.shape[0], "train samples")
print(X_test.shape[0], "test samples")

Y_train = np_utils.to_categorical(y_train,nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

i =26
plt.imshow(X_train[i,0], interpolation = "nearest")
print ("label :", Y_train[i,:])
```



```
model = Sequential()
model.add(Convolution2D(nb_filters,nb_conv, nb_conv, border_mode= "valid", input_shape = (1,img_rows, img_cols)
convout1 = Activation("relu")
model.add(convout1)
model.add(Convolution2D(nb_filters,nb_conv, nb_conv))
convout2 = Activation("relu")
model.add(convout2)
model.add(MaxPooling2D(pool_size= (nb_pool, nb_pool)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128))
model.add(Activation("relu"))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation("softmax"))
model.compile(loss = "categorical_crossentropy",optimizer= "adadelta",metrics=["accuracy"])
```



```
#%%

model.fit(X_train, Y_train, batch_size= batch_size, nb_epoch= nb_epoch, show_accuracy = True, verbose= 1, valid

model.fit(X_train, Y_train, batch_size= batch_size, nb_epoch= nb_epoch, show_accuracy = True, verbose= 1, valid
```



```
#%%
score = model.evaluate(X_test, Y_test, show_accuracy =True, verbose= 0)

print('Test score:', score[0])
print('test_Accuracy:', score[1])
print(model.predict_classes(X_test[1:5]))
print(Y_test[1:5])
```

Diabetic Retinopathy Debrecen Data Set

- **Data Set Information:**

- This dataset contains features extracted from the Messidor image set to predict whether an image contains signs of diabetic retinopathy or not. All features represent either a detected lesion, a descriptive feature of a anatomical part or an image-level descriptor.

- **Attribute Information:**

- 1) The binary result of pre-screening, where 1 indicates severe retinal abnormality and 0 its lack.
- 2-7) The results of MA detection. Each feature value stand for the number of MAs found at the confidence levels alpha = 0.5, . . . , 1, respectively.
- 8-15) contain the same information as 2-7) for exudates. However, as exudates are represented by a set of points rather than the number of pixels constructing the lesions, these features are normalized by dividing the number of lesions with the diameter of the ROI to compensate different image sizes.
- 16) The Euclidean distance of the center of the macula and the center of the optic disc to provide important information regarding the patients condition.
- 17) The diameter of the optic disc.
- 18) The binary result of the AM/FM-based classification.
- 19) Class label. 1 = contains signs of DR (Accumulative label for the Messidor classes 1, 2, 3), 0 = no signs of DR.

Logistic Regression:

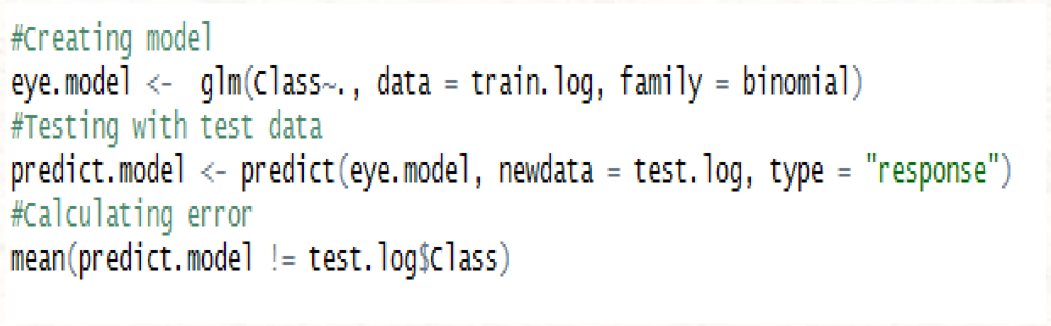
- ➢ When and Why

- To predict an outcome variable that is categorical from predictor variables that are continuous and/or categorical
- Logistic regression deals with this problem by using a logarithmic transformation on the outcome variable which allow us to model a nonlinear association in a linear way.
- It expresses the linear regression equation in logarithmic terms (called the logit)

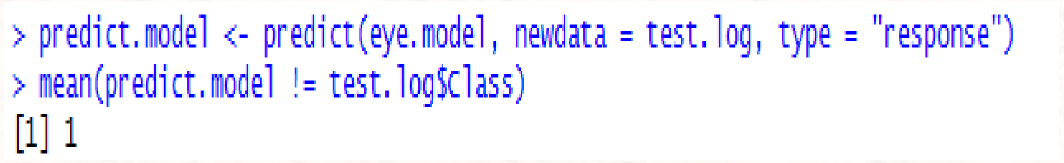
- ➢ It helps to determine:

- relative importance of each predictor
- Are there interactions among predictors
- How good is the model at classifying cases for which the outcome is known

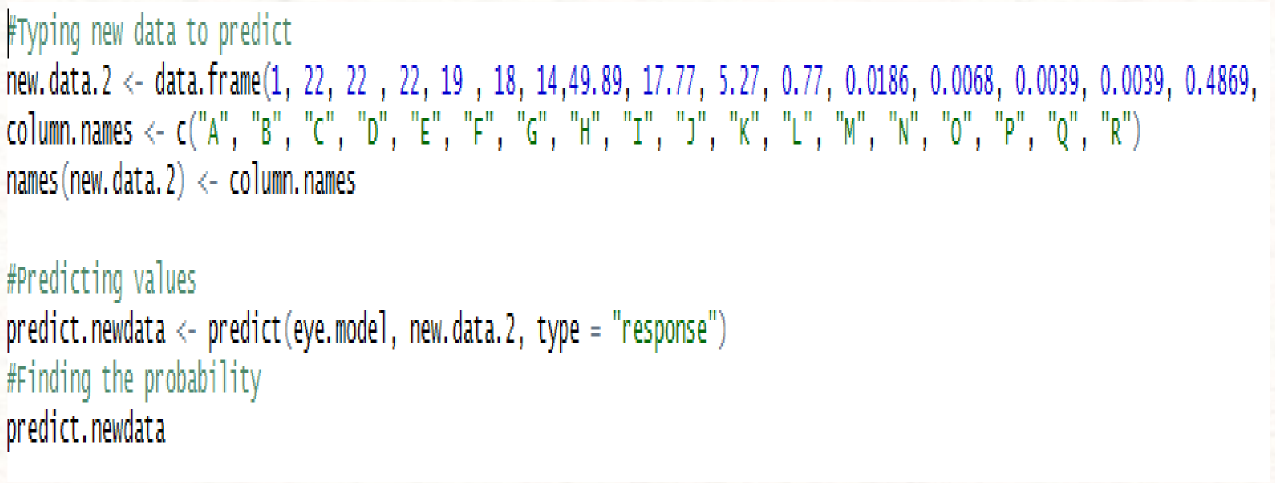
Logistic Regression:



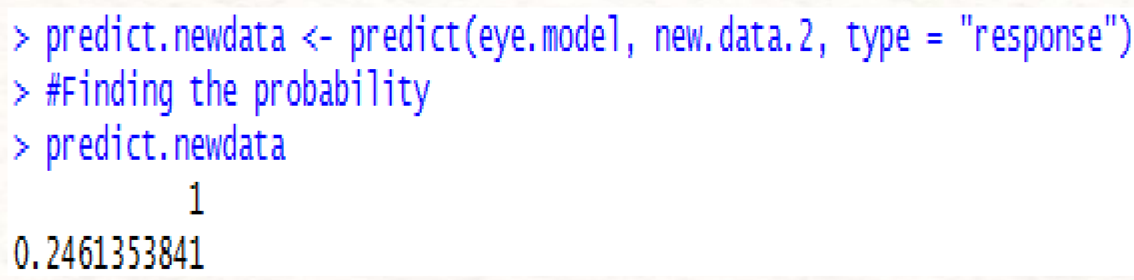
```
#Creating model
eye.model <-  glm(Class~., data = train.log, family = binomial)
#Testing with test data
predict.model <- predict(eye.model, newdata = test.log, type = "response")
#Calculating error
mean(predict.model != test.log$Class)
```



```
> predict.model <- predict(eye.model, newdata = test.log, type = "response")
> mean(predict.model != test.log$Class)
[1] 1
```



```
#Typing new data to predict
new.data.2 <- data.frame(1, 22, 22 , 22, 19 , 18, 14,49.89, 17.77, 5.27, 0.77, 0.0186, 0.0068, 0.0039, 0.0039, 0.4869,
column.names <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R")
names(new.data.2) <- column.names

#Predicting values
predict.newdata <- predict(eye.model, new.data.2, type = "response")
#Finding the probability
predict.newdata
```

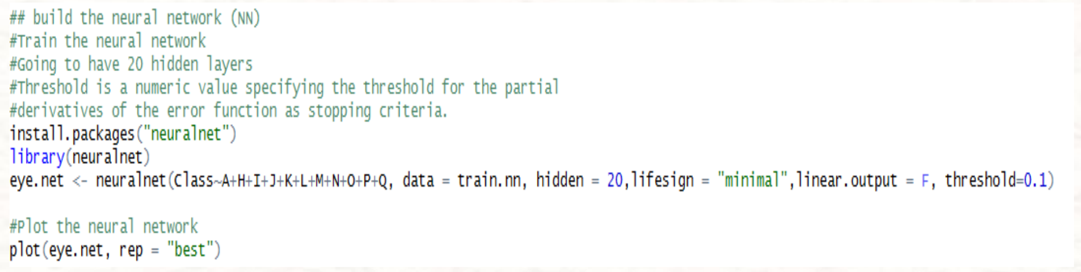


```
> predict.newdata <- predict(eye.model, new.data.2, type = "response")
> #Finding the probability
> predict.newdata
         1
0.2461353841
```

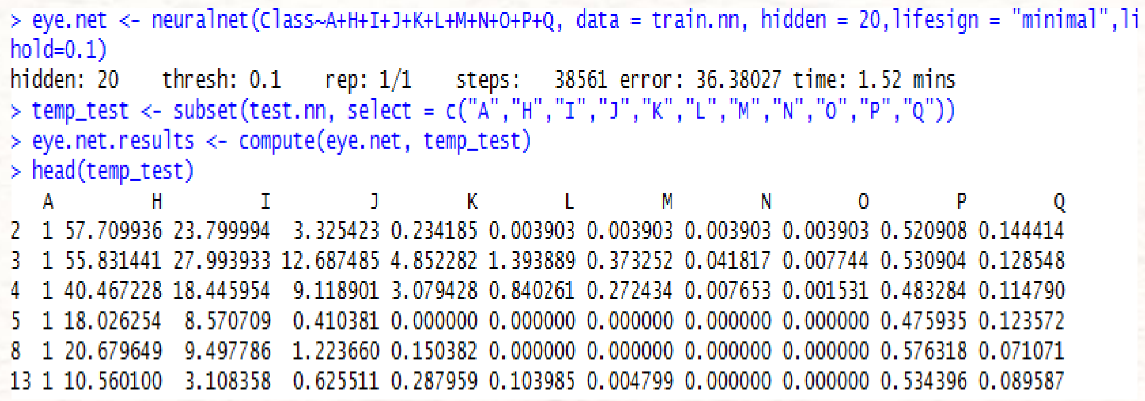
Neural Network

- Neural networks are situated in the domain of machine learning.
- The dataset will be split up in a subset used for training the neural network and another set used for testing. As the ordering of the dataset is completely random.
- We'll build a neural network with 20 hidden nodes (a neural network is comprised of a input, hidden and output nodes)
- The output is not linear and we will use a threshold value of 10%.
- The neural net package uses weight backtracking as its standard algorithm

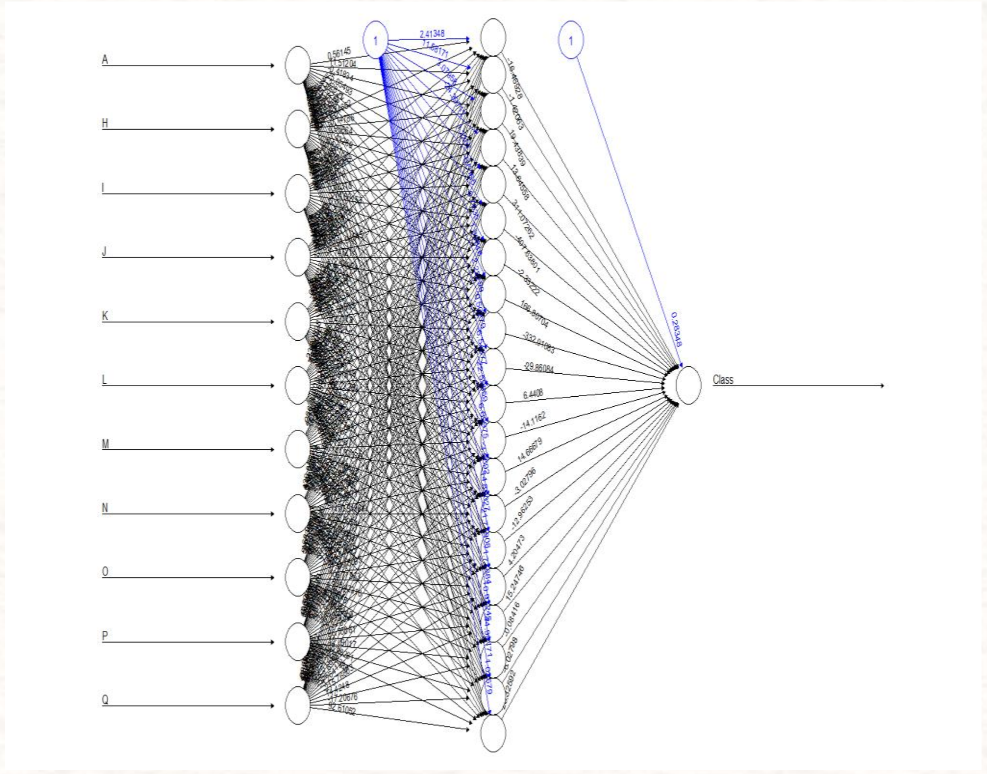
Neural Network:

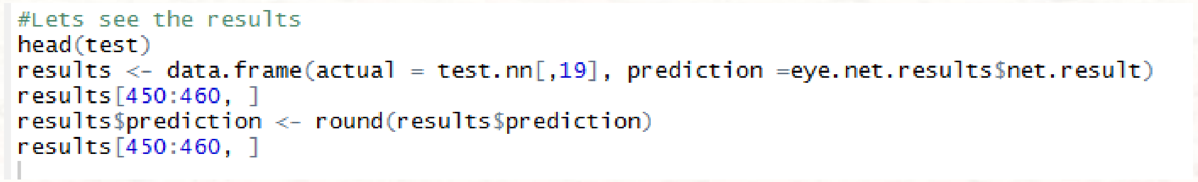


```
## build the neural network (NN)
#Train the neural network
#Going to have 20 hidden layers
#Threshold is a numeric value specifying the threshold for the partial
#derivatives of the error function as stopping criteria.
install.packages("neuralnet")
library(neuralnet)
eye.net <- neuralnet(Class~A+H+I+J+K+L+M+N+O+P+Q, data = train.nn, hidden = 20,lifesign = "minimal",linear.output = F, threshold=0.1)

#Plot the neural network
plot(eye.net, rep = "best")
```

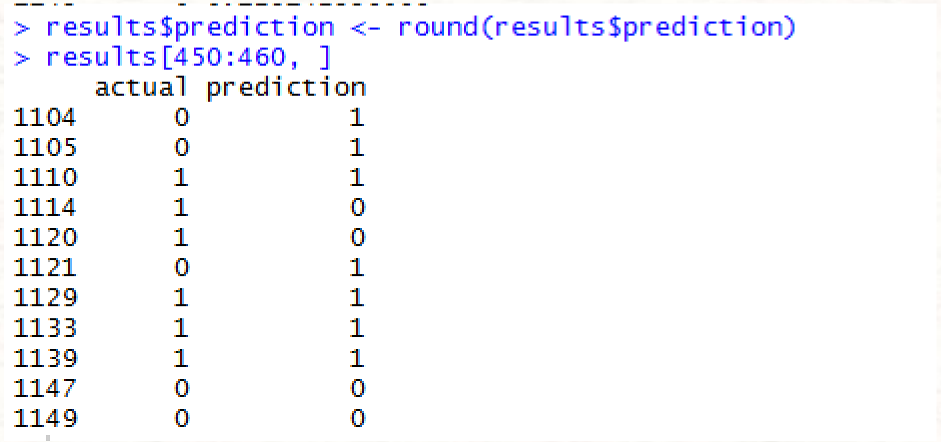


```
> eye.net <- neuralnet(Class~A+H+I+J+K+L+M+N+O+P+Q, data = train.nn, hidden = 20,lifesign = "minimal",li
hold=0.1)
hidden: 20    thresh: 0.1    rep: 1/1    steps:   38561 error: 36.38027 time: 1.52 mins
> temp_test <- subset(test.nn, select = c("A","H","I","J","K","L","M","N","O","P","Q"))
> eye.net.results <- compute(eye.net, temp_test)
> head(temp_test)
   A         H         I         J        K        L        M        N        O        P        Q
2  1 57.709936 23.799994  3.325423 0.234185 0.003903 0.003903 0.003903 0.003903 0.520908 0.144414
3  1 55.831441 27.993933 12.687485 4.852282 1.393889 0.373252 0.041817 0.007744 0.530904 0.128548
4  1 40.467228 18.445954  9.118901 3.079428 0.840261 0.272434 0.007653 0.001531 0.483284 0.114790
5  1 18.026254  8.570709  0.410381 0.000000 0.000000 0.000000 0.000000 0.000000 0.475935 0.123572
8  1 20.679649  9.497786  1.223660 0.150382 0.000000 0.000000 0.000000 0.000000 0.576318 0.071071
13 1 10.560100  3.108358  0.625511 0.287959 0.103985 0.004799 0.000000 0.000000 0.534396 0.089587
```





```
#Lets see the results
head(test)
results <- data.frame(actual = test.nn[,19], prediction =eye.net.results$net.result)
results[450:460, ]
results$prediction <- round(results$prediction)
results[450:460, ]
```



```
> results$prediction <- round(results$prediction)
> results[450:460, ]
     actual prediction
1104      0          1
1105      0          1
1110      1          1
1114      1          0
1120      1          0
1121      0          1
1129      1          1
1133      1          1
1139      1          1
1147      0          0
1149      0          0
```

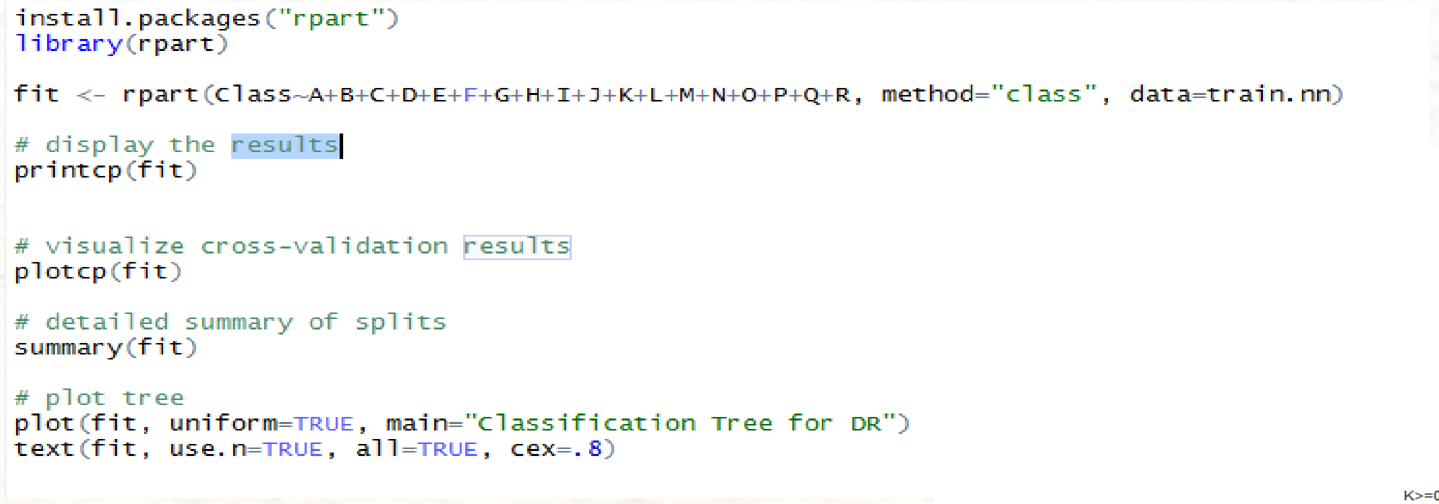
Classification Tree (CART):

- Tree methods such as CART (classification and regression trees) can be used as alternatives to logistic regression. It is a way that can be used to show the probability of being in any hierarchical group.
- The concept of trees and forests can be applied in many different setting and is often seen in machine learning and data mining settings or other settings where there is a significant amount of data.
- (rpart):This package includes several example sets of data that can be used for recursive partitioning and regression trees.

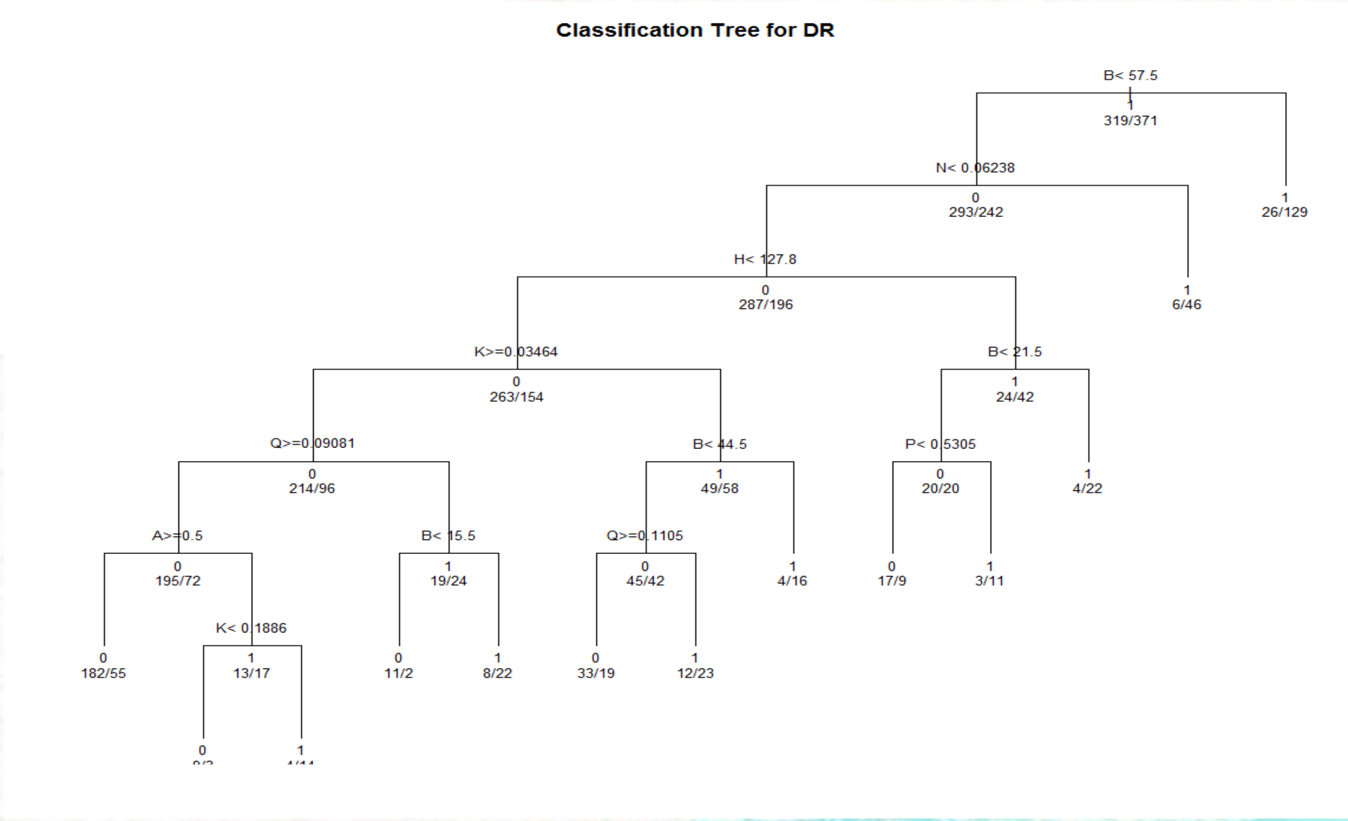
➢ Advantages:

- **Decision trees implicitly perform variable screening or feature selection**
- **Decision trees require relatively little effort from users for data preparation**
- **Nonlinear relationships between parameters do not affect tree performance**
- **easy to interpret and explain**

Classification Tree (CART):



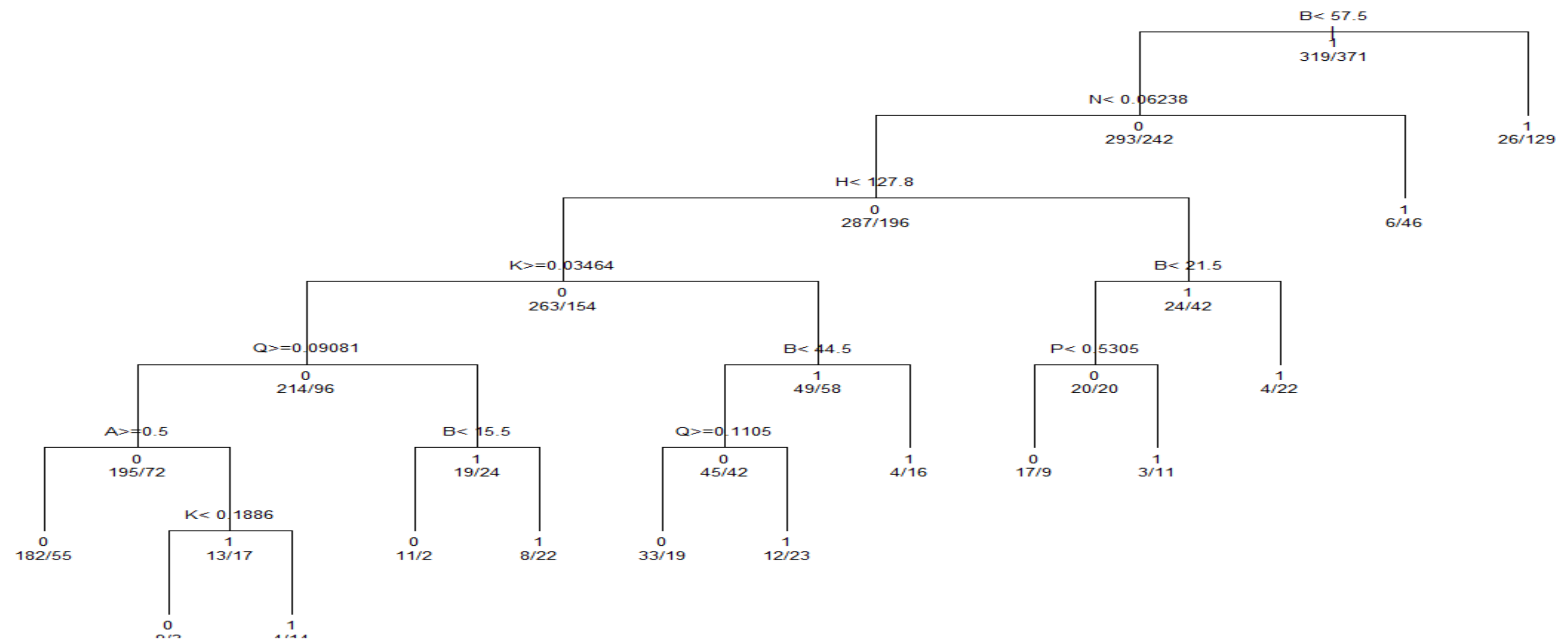
```
install.packages("rpart")
library(rpart)

fit <- rpart(Class~A+B+C+D+E+F+G+H+I+J+K+L+M+N+O+P+Q+R, method="class", data=train.nn)

# display the results
printcp(fit)


# visualize cross-validation results
plotcp(fit)

# detailed summary of splits
summary(fit)

# plot tree
plot(fit, uniform=TRUE, main="Classification Tree for DR")
text(fit, use.n=TRUE, all=TRUE, cex=.8)
```



```r
#Typing new data to predict
new.data.2 <- data.frame(1, 22, 22 , 22, 19 , 18, 14,49.89, 17.77, 5.27, 0.77, 0.0186, 0.0068, 0.0039, 0.0039, 0.4869, 0.100, 1)
column.names <- c("A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R")
names(new.data.2) <- column.names
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	Class
1	22	22	22	19	18	14	49.89576	17.77599	5.27092	0.771761	0.018632	0.006864	0.003923	0.003923	0.486903	0.100025	1	0

Classification Tree for DR



KNN

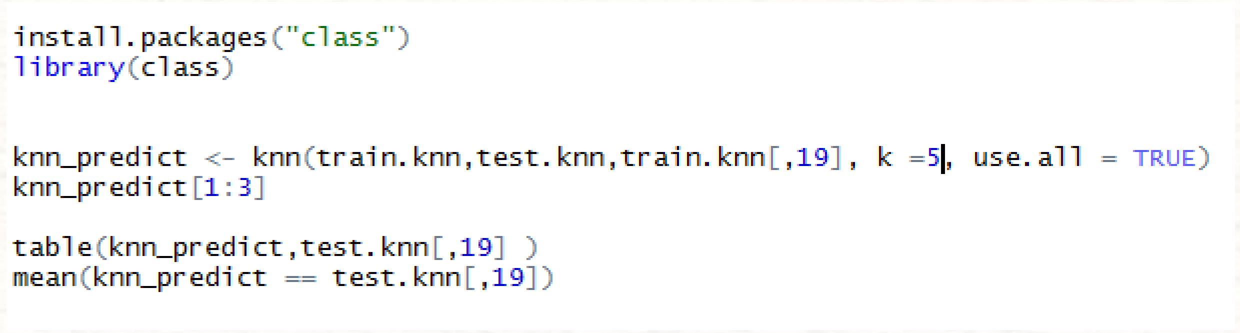
- ➢ k-Nearest Neighbors (k-NN)

- A simple supervised learning algorithm is k-Nearest Neighbors algorithm (k-NN). KNN is a non-parametric method used for classification and regression.
- In both cases, the input consists of the k closest training examples in the feature space.

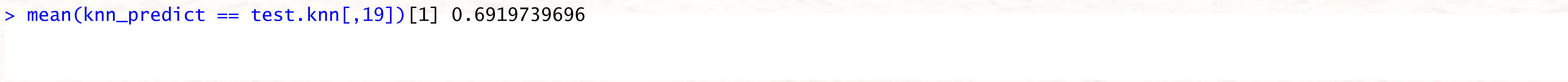
- ➢ Nearest neighbor search

A simple solution to finding nearest neighbors is to compute the distance from the each point in S to every point in R and keeping track of the "best so far". This algorithm,sometimes referred to as the naive approach, has a running time of O(|R||S|).

KNN:



```
install.packages("class")
library(class)


knn_predict <- knn(train.knn,test.knn,train.knn[,19], k =5, use.all = TRUE)
knn_predict[1:3]

table(knn_predict,test.knn[,19] )
mean(knn_predict == test.knn[,19])
```



```
> mean(knn_predict == test.knn[,19])[1] 0.6919739696
```

Reference:

- https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set#
- https://www.kaggle.com/c/diabetic-retinopathy-detection/data
- https://www.r-bloggers.com/using-neural-networks-for-credit-scoring-a-simple-example/
- https://www.kaggle.com/c/diabetic-retinopathy-detection
- https://en.wikipedia.org/wiki/Diabetic_retinopathy

THANK YOU