# Chapter 04 Discriminant Analysis Multiclass classifier

### Discriminant Analysis (LDA, QDA)

Wiki is a good place to start https://en.wikipedia.org/wiki/Linear_discriminant_analysis
There are several other useful sources. I found Kardi tutorial to be very helpful
https://people.revoledu.com/kardi/tutorial/LDA/LDA%20Formula.htm.

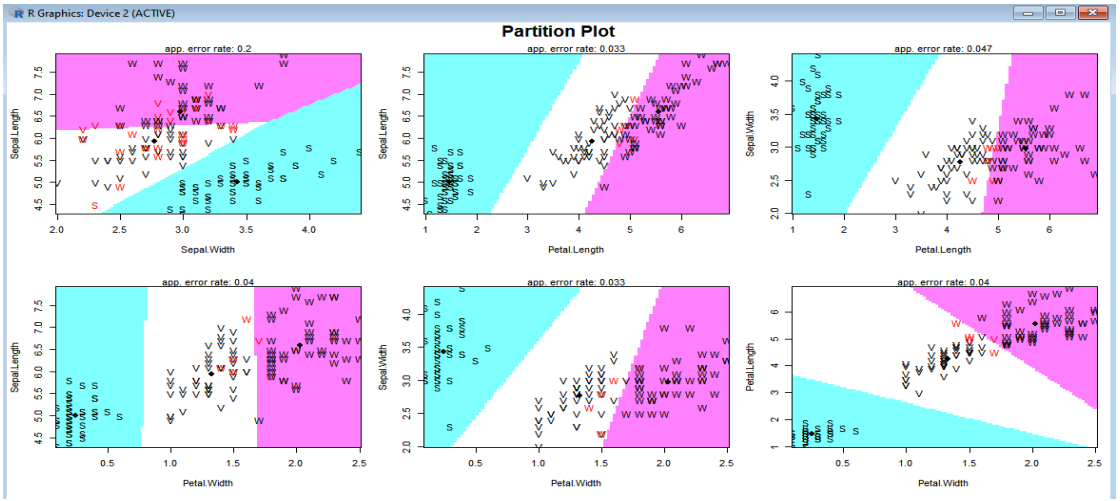
Assume that the requirement is to identify the class (one of K classes)given a feature vector X.
partimat from klaR package helps us visualize the discriminating capacity of pairs of features.
Today we will start with klaR package to visualize datasets, in particular, how the features segment the class variable. Let us consider iris dataset that comes preloaded in R.

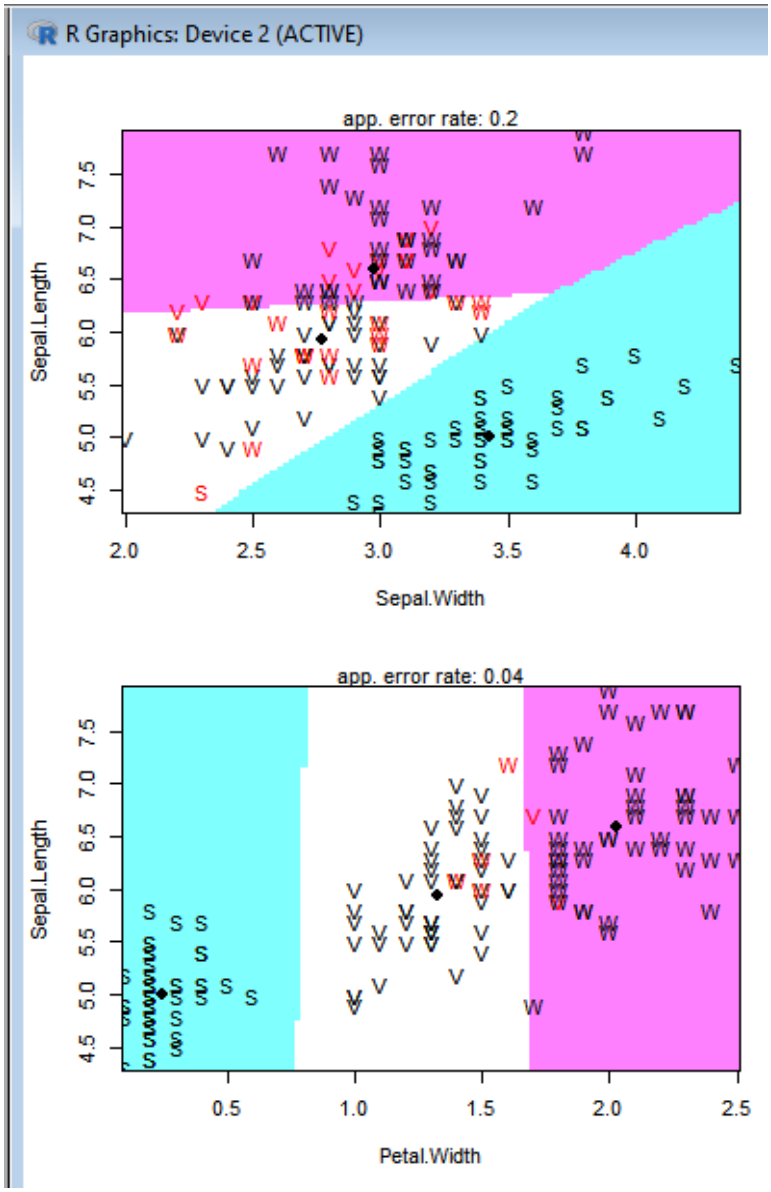
```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
```

iris has 5 columns, the Species column being the class column and the other four columns are features. We want to examine the dataset to understand how these features segment the class variable. Is there an obvious decision boundary – class separation. For example below X < Z, class is C1 and so on.

We use klaR package to visualiize such relationships.

*if (!require(klaR)) install.packages('klaR')*
*library('klaR')*



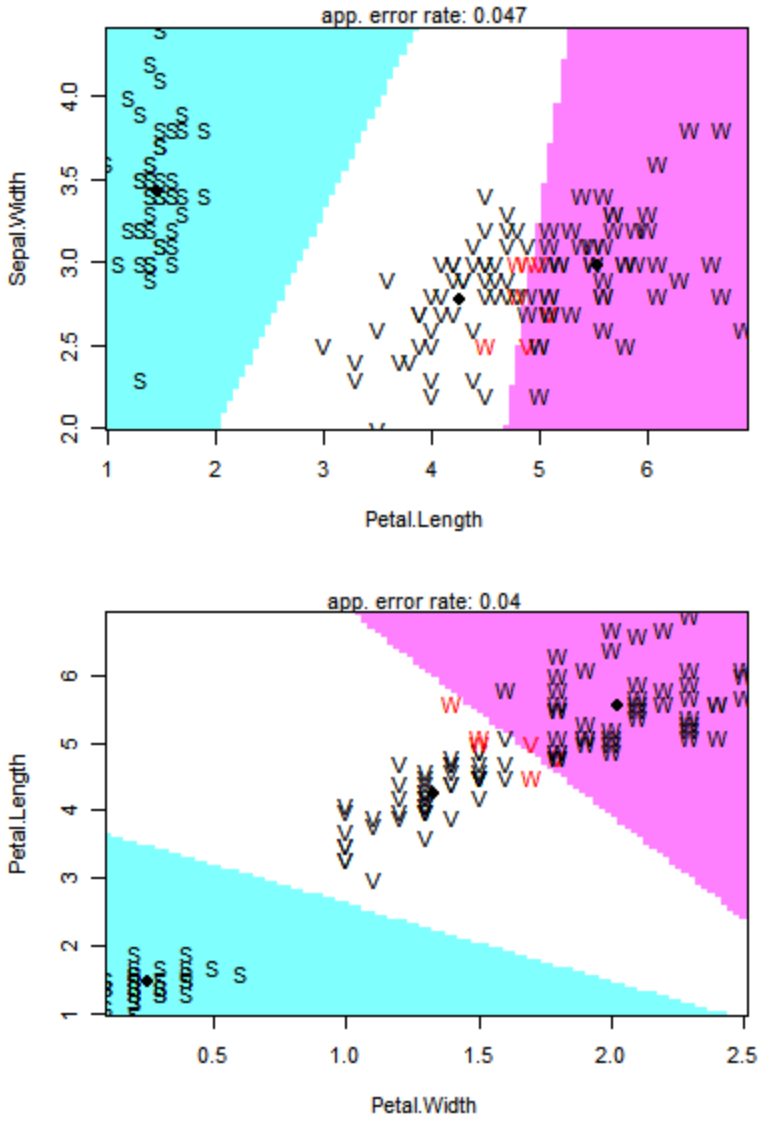



we start with

partimat(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,data=iris,method="lda")

But two classes are represented by the same letter V so let us change it to remove the ambiguity.

```
iris.lda<-iris # let us make a copy and change the species names
iris.lda$Species<-ifelse(iris$Species=='virginica','Wirginica',
 ifelse(iris$Species=='versicolor','VERSI','SETOSA'))
# we expect to see S,V and W in the regions
table(iris.lda$Species)
iris.lda$Species<-as.factor(iris.lda$Species)
partimat(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,data=iris.lda,method="lda")
```




Here I have used the partimat function from klaR package over iris dataset as shown above.

*Let us use Fisher's Linear Discriminant Analysis to classify the iris dataset.*

Now let us try to understand one feature at a time and its relationship with Class variable using ISLR 4th chapter equations: I have attached scan of the important equations below. This book is freely available on the internet.

the parameters $\mu_1, \ldots, \mu_K$, $\pi_1, \ldots, \pi_K$, and $\sigma^2$. The *linear discriminant*



*analysis* (LDA) method approximates the Bayes classifier by plugging estimates for $\pi_k$, $\mu_k$, and $\sigma^2$ into (4.13). In particular, the following estimates are used:

linear discriminant analysis

$$\begin{aligned}\hat{\mu}_k &= \frac{1}{n_k}\sum_{i:y_i=k} x_i \\ \hat{\sigma}^2 &= \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:y_i=k}(x_i-\hat{\mu}_k)^2 \end{aligned} \tag{4.15}$$

where $n$ is the total number of training observations, and $n_k$ is the number of training observations in the $k$th class. The estimate for $\mu_k$ is simply the average of all the training observations from the $k$th class, while $\hat{\sigma}^2$ can be seen as a weighted average of the sample variances for each of the $K$ classes. Sometimes we have knowledge of the class membership probabilities $\pi_1, \ldots, \pi_K$, which can be used directly. In the absence of any additional information, LDA estimates $\pi_k$ using the proportion of the training observations that belong to the $k$th class. In other words,

$$\hat{\pi}_k = n_k/n. \tag{4.16}$$

The LDA classifier plugs the estimates given in (4.15) and (4.16) into (4.13), and assigns an observation $X = x$ to the class for which

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k) \tag{4.17}$$

is largest. The word *linear* in the classifier's name stems from the fact that the *discriminant functions* $\hat{\delta}_k(x)$ in (4.17) are linear functions of $x$ (as opposed to a more complex function of $x$).

discriminant function

```
pi_lda <- function(y){ #
Equation 4.16
 pi_est <- table(y) / length(y)
 return(as.matrix(pi_est))
} # class proportion
```

```
mu_lda <- function(X, y){ # Equation 4.15, for mu
 data_est <- as.data.frame(cbind(X,y))
 data_est$X <- as.numeric(as.character(data_est$X))
mu <- aggregate(data = data_est, X ~ y, FUN = "mean")
 colnames(mu) <- c("y", "X")
 return(mu) # class wise mean
}
```

```
var_lda <- function(X, y, mu){ # Equation 4.15, for sigma**2
  n <- length(X)
  K <- length(unique(y))
  k <- unique(y)
  var_est <- 0
  for (i in 1:K){
    var_est <- sum((X[y == k[i]] - mu$X[k[i] == mu$y])^2) + var_est
#X[[X[y=='setosa']-mu$X[setosa==mu$y
  }
  var_est <- (1 / (n - K)) * var_est
  return(var_est)
}
```

```
discriminant_lda <- function(X, pi, mu, var){# Equation 4.17
  K <- length(unique(y))
  k <- unique(y)
  disc <- matrix(nrow = length(X), ncol = K)
  colnames(disc) <- k
    for (i in 1:K){
    disc[ ,i] <- X * (mu$X[i] / var) - ((mu$X[i]^2) / (2 * var)) + log(pi[i])
  }
  disc <- as.data.frame(disc)
  disc$predict <- apply(disc, 1, FUN = "which.max")
  return(disc)
}
# the Species is the target variable, 5th column in iris
y <- as.character(iris[ ,5])
 pi_est <- pi_lda(y)
```

```
> pi_est
                 [,1]
setosa     0.3333333
versicolor 0.3333333
virginica  0.3333333
>
```

LDA makes the assumption the variances are the same across the feature vectors.
Let us run through the feature vector, one by one, and estimate their ability to predict using the mean and variance of a that feature.

```
X <- iris[ ,1] # we are reducing our feature space to be univariate
```

*mu_est_1 <- mu_lda(X, y)*

*var_est_1 <- var_lda(X, y, mu_est_1)*

*discriminant_est_1 <- discriminant_lda(X, pi_est, mu_est_1, var_est_1)*

```
table( iris$Species,discriminant_est_1$predict)
```

```
> table( iris$Species,discriminant_est_1$predict)

               1  2  3
  setosa      45  5  0
  versicolor   6 30 14
  virginica    1 12 37
>
```

```
X <- iris[ ,2]
 mu_est_2 <- mu_lda(X, y)
var_est_2 <- var_lda(X, y, mu_est_2)
discriminant_est_2 <- discriminant_lda(X, pi_est, mu_est_2, var_est_2)
table( iris$Species,discriminant_est_2$predict)
```

```
> table( iris$Species,discriminant_est_2$predict)

               1  2  3
  setosa      33  1 16
  versicolor   2 27 21
  virginica    8 19 23
>
```

```
 X <- iris[ ,3]
 mu_est_3 <- mu_lda(X, y)
 var_est_3 <- var_lda(X, y, mu_est_3)
 discriminant_est_3 <- discriminant_lda(X, pi_est, mu_est_3, var_est_3)
 table( iris$Species,discriminant_est_3$predict)
```

```
> table( iris$Species,discriminant_est_3$predict)

               1  2  3
  setosa      50  0  0
  versicolor   0 48  2
  virginica    0  6 44
```

```
X <- iris[ ,4]
mu_est_4 <- mu_lda(X, y)
var_est_4 <- var_lda(X, y, mu_est_4)
discriminant_est_4 <- discriminant_lda(X, pi_est, mu_est_4, var_est_4)
 table( iris$Species,discriminant_est_4$predict)
```

```
> table( iris$Species,discriminant_est_4$predict)

               1  2  3
  setosa      50  0  0
  versicolor   0 48  2
  virginica    0  4 46
```

The best performance (or lowest error) we get is 6 misclassifications

# can there be a meta learner to combine these single feature classifier

*predictions<-data.frame(actual=iris$Species,pred1=discriminant_est_1$predict,*
*pred2=discriminant_est_2$predict,pred3=discriminant_est_3$predict,*
*pred4=discriminant_est_4$predict)*
*predictions*

```
> sqldf('select * from predictions where pred3 <> pred4 ')
       actual pred1 pred2 pred3 pred4
1  versicolor     2     3     2     3
2  versicolor     2     2     3     2
3   virginica     1     2     2     3
4   virginica     2     2     3     2
5   virginica     2     2     2     3
6   virginica     3     2     2     3
7   virginica     2     2     2     3
8   virginica     2     3     2     3
9   virginica     3     3     3     2
10  virginica     3     2     3     2
11  virginica     2     2     3     2
12  virginica     2     3     2     3
```

We can see that a single attribute can detect species as well as it does, atleast for the iris dataset.

Let us now consider an implementation of ..equation ISLR 4.19 from implemented by Professor Dalal (Columbia.edu).

bit of algebra reveals that the Bayes classifier assigns an observation $X = x$ to the class for which

$$\delta_k(x) = x^T \mathbf{\Sigma}^{-1} \mu_k - \frac{1}{2} \mu_k^T \mathbf{\Sigma}^{-1} \mu_k + \log \pi_k \qquad (4.19)$$

is largest. This is the vector/matrix version of (4.13).

*# Let us setup three distinct data.frames*
*default_setosa<-iris[iris[,5]=='setosa',]*
*default_virginica<-iris[iris[,5]=='virginica',]*
*default_versicolor<-iris[iris[,5]=='versicolor',]*

*# for each class, let us compute the mean for each feature*
*mu_setosa<-apply(default_setosa[,1:4],2,mean)*

```
mu_virginica<-apply(default_virginica[,1:4],2,mean)
mu_versicolor<-apply(default_versicolor[,1:4],2,mean)
# for each class, let us compute the cov
sigma_setosa<-cov(default_setosa[,1:4])
sigma_virginica<-cov(default_virginica[,1:4])
sigma_versicolor<-cov(default_versicolor[,1:4])
# for each class, get the number of rows
n_setosa<-dim(default_setosa)[1]
n_virginica<-dim(default_virginica)[1]
n_versicolor<-dim(default_versicolor)[1]
# let us compute the pooled covariance
sigma_all<-((n_setosa-1)*sigma_setosa+(n_virginica-1)*sigma_virginica+(n_versicolor-
1)*sigma_versicolor)/(n_setosa+n_virginica+n_versicolor-3) #pooled Cov matrix-
# a vector of means
mu<-cbind(mu_setosa,mu_virginica,mu_versicolor)
# prior probabilities for each class
pi.vec <- rep(0,3) # why not just pi?
pi.vec[1] <- sum((iris$Species=='setosa'))/nrow(iris)
pi.vec[2] <- sum((iris$Species=='virginica'))/nrow(iris)
pi.vec[3] <- sum((iris$Species=='versicolor'))/nrow(iris)
# the discriminant function implementing 4.19 from ISLR
my.lda<-function(pi.vec,mu,Sigma,x){
x.dims <- dim(x)
n <- x.dims[1]
Sigma.inv <- solve(Sigma) #Find inverse of Sigma
out.prod <- rep(1,n) #all items initiated to be negative
# equation 4.19 from ISLR Sixth Printing Page 157
discrim.setosa <- apply(x,1,function(y) y %*% Sigma.inv %*% mu[,1]
- 0.5*t(mu[,1]) %*% Sigma.inv %*% mu[,1] + log(pi.vec[1]))
discrim.virginica <- apply(x,1,function(y) y %*% Sigma.inv %*% mu[,2]
- 0.5*t(mu[,2]) %*% Sigma.inv %*% mu[,2] + log(pi.vec[2]))
discrim.versicolor <- apply(x,1,function(y) y %*% Sigma.inv %*% mu[,3]
- 0.5*t(mu[,3]) %*% Sigma.inv %*% mu[,3] + log(pi.vec[3]))
probs<-data.frame(p1=discrim.setosa,p2=discrim.virginica,p3=discrim.versicolor)
out.prod<-unlist(apply(probs,1,which.max))
return(out.prod)
}
pred_default_all<-my.lda(pi.vec,mu,sigma_all,iris[,1:4])
```

```
> pred_species<-ifelse(pred_default_all==1,'setosa',ifelse(pred_default_all=='3',as.character(iris[51,5]),
+ as.character(iris[101,5])))
> table(iris[,5],pred_species)
            pred_species
             setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          1        49
```

Performance (accuracy) has increased with our MV implmentation. We now have only 3 misclassifications. In contrast, with univariate model, the lowest misclassifications we achieved was 6.

Let us compute the other metrics, precision,recall,specificity,sensitivity, using ***caret*** package.

```
> caret::confusionMatrix(table(iris[,5],pred_species))
Confusion Matrix and Statistics

            pred_species
             setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         48         2
  virginica       0          1        49

Overall Statistics

               Accuracy : 0.98
                 95% CI : (0.9427, 0.9959)
    No Information Rate : 0.34
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.97

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.9796           0.9608
Specificity                 1.0000            0.9802           0.9899
Pos Pred Value              1.0000            0.9600           0.9800
Neg Pred Value              1.0000            0.9900           0.9800
Prevalence                  0.3333            0.3267           0.3400
Detection Rate              0.3333            0.3200           0.3267
Detection Prevalence        0.3333            0.3333           0.3333
Balanced Accuracy           1.0000            0.9799           0.9753
```

This concludes our exploration of parametric classifiers.

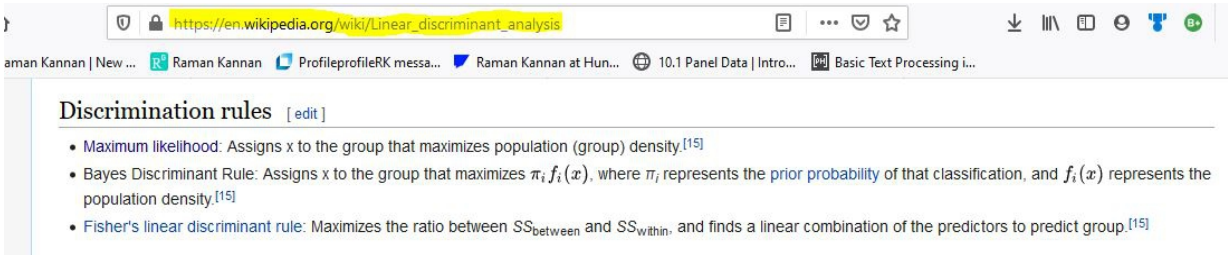
Concluding Remarks

These are supervised learners – first they learn using samples with labels , during the induction or learning phase and then during the generalization phase (aka scoring) label a unlabeled observation when presented. Logistic,NB and LDA all assume that data is distributed according to some known distribution. A normal distribution is fully defined by the mean and the variance. Most other common distributions are defined by a few parameters. Hence, these are called parametric methods. Logistic is binary classifier (1 and 0) and seeks a linear seperation boundary. NaiveBayes and LDA are multiclass. – LDA is linear classifier because 4.17 is linear in x. Here we explained using iris dataset. In the RMD we will implement using heart dataset.

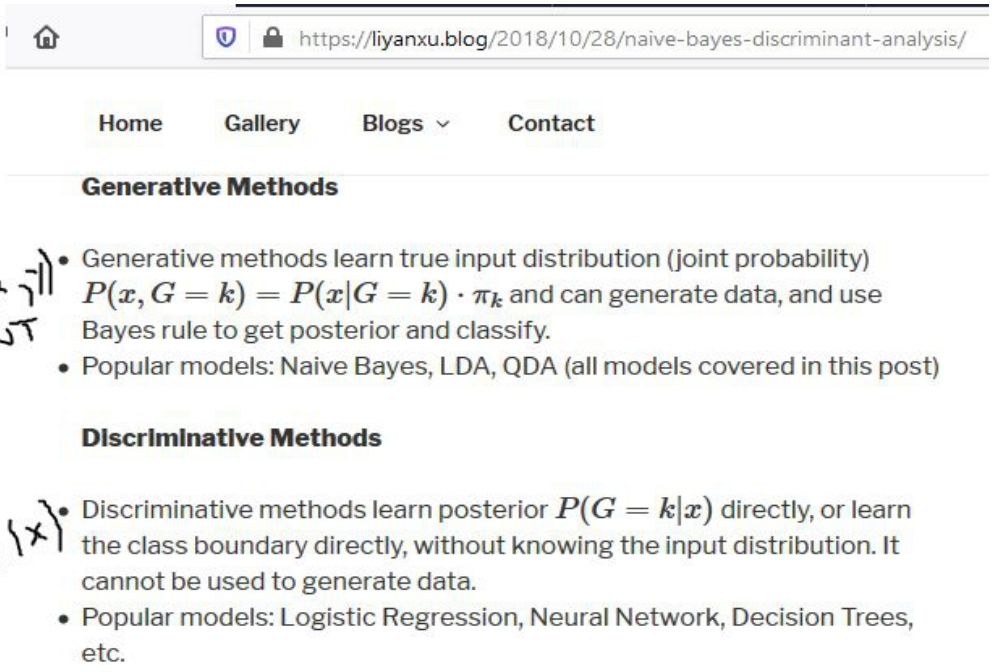
Logistic Regression transforms the features using logit (sigmoid function) and estimates the betas using Gradient Descent or MLE or other optimization techniques.

NaiveBayes assumes features are class-conditionally independent and is easily derived from Bayes theorem. All three parametric methods we have covered are Bayesian Estimators in that they assign the class with the highest probability.

A wikipedia offers a crisp definition of these approaches




It will not be complete review of parametric classifiers without mentioning the two families of parametric estimators – generative and discriminative, a brief review follows. We can reason about Parametric classifiers whether they are computing joint probabilities or conditional probabilities. Classifiers that compute p(x,y) are generative – as you can generate new observations given p(x,y). On the other hand classifiers that compute p(x|y) are discriminative.




Another useful definition is based on what the classifier does. Does it determine a boundary that separates one class from the other. If it does it is a discriminative classifier. Given a decision boundary we cannot generate new observations. It is good to know the lingo and to appreciate p(x|y) is easier than computing p(x,y). NaiveBayes is considered generative and Logistic Regression is considered discriminative.

We will now go onto evaluate non-parametric methods namely kNN – k Nearest Neighbor and Decision Tree and support vector machines . *Practice alone makes perfect.*