Homework 3

Gavin Monroe - ComS 474

1) **Suppose that we wish to predict whether a given stock will issue a dividend this year ("Yes" or "No") based on X, last year's percent profit. We examine a large number of companies and discover that the mean value of X for companies that issued a dividend was X¯ = 10, while the mean for those that didn't was X¯ = 0. In addition, the variance of X for these two sets of companies was ˆσ2 = 36. Finally, 80 % of companies issued dividends. Assuming that X follows a normal distribution, predict the probability that a company will issue a dividend this year given that its percentage profit was X = 4 last year. Hint: Recall that the density function for a normal random variable**

It suffices to plug in the parameters and X values in the equation for pk(x). We get

$$p_1(4) = \frac{0.8e^{-(1/72)(4-10)^2}}{0.8e^{-(1/72)(4-10)^2} + 0.2e^{-(1/72)(4-0)^2}} = 0.752;$$

so the probability that a company will issue a dividend this year given that its percentage return was X=4 last year is 0.752.

2) Hint: Read Section 4.2.0

Suppose you are predicting a feature y that can take on three values y ∈ {+1, +2, +3} and you can predict y using two features x1 and x2. You decide to try LDA. Suppose that the covariance matrices for x1 and x2 are the identity matrix,

Σy=+1 = Σy=+2 = Σy=+3 = [1 0, 0 1] ,

the priors are equal

πy=+1 = πy=+2 = πy=+3 = 1 3 ,

and the mean vectors are

μy=+1 = −1 −1 μy=+2 = 1 1 μy=+3 = [−1, 1]

(a). Find the equation for the LDA boundary between y = +1 and y = +2.

Using baye's theorem for the following we can plug them into the following equation

1. $p(y = 1 \mid x) = \frac{p(x|y=1)\pi_1}{p(x)}$
2. $p(y = 2 \mid x) = \frac{p(x|y=2)\pi_2}{p(x)}$.

Plugging in a two-dimensional input x into these formulas, you will get the probability of

ending up with the conditional probability that y=1 or y=2. The decision boundary is the line in $\mathbb{R}2$ where these two conditional probabilities are equal. To find this line, solve
$p(y=1|x)=p(y=2|x)$ or
$p(x|y=1)\pi 1=p(x|y=2)\pi 2$.
Taking it a little further:
$.33\exp(-[x-\mu 1]T\Sigma-1[x-\mu 1])-.33\exp(-[x-\mu 2]T\Sigma-1[x-\mu 2])=0$
$\exp(-\{-xT\Sigma-1\mu 1+xT\Sigma-1\mu 2+\mu T1\Sigma-1\mu 1-\mu T2\Sigma-1\mu 2\})$=>
$-xT\Sigma-1\mu 1+xT\Sigma-1\mu 2+\mu T1\Sigma-1\mu 1-\mu T2\Sigma-1\mu 2$=>
$xT[2\Sigma-1(\mu 2-\mu 1)]+\{\mu T1\Sigma-1\mu 1-\mu T2\Sigma-1\mu 2\}$=>

-1 -x1 - x2 = -1 + x1 + x2 => 0

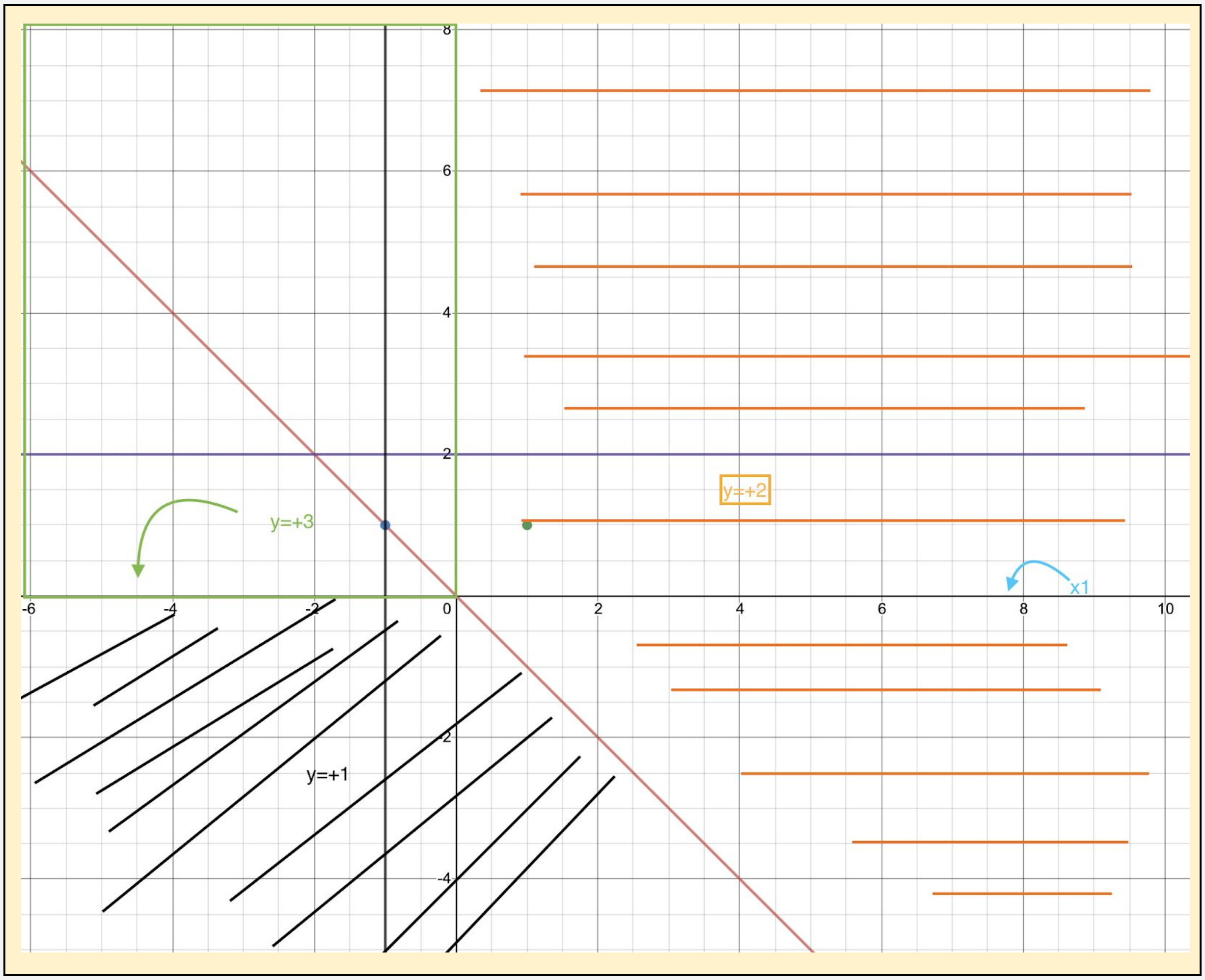
(b). Find the equation for the LDA boundary between y = +1 and y = +3.

$.33\exp(-[x-\mu 1]T\Sigma-1[x-\mu 1])-.33\exp(-[x-\mu 2]T\Sigma-1[x-\mu 2])=0$
$\exp(-\{-xT\Sigma-1\mu 1+xT\Sigma-1\mu 2+\mu T1\Sigma-1\mu 1-\mu T2\Sigma-1\mu 2\})$=>
$-xT\Sigma-1\mu 1+xT\Sigma-1\mu 2+\mu T1\Sigma-1\mu 1-\mu T2\Sigma-1\mu 2$=>
$xT[3\Sigma-1(\mu 2-\mu 1)]+\{\mu T1\Sigma-1\mu 1-\mu T2\Sigma-1\mu 2\}$=>
2x2 = 0 => x2 => 0 -> decision boundary

(c). Find the equation for the LDA boundary between y = +2 and y = +3.

$.33\exp(-[x-\mu 1]T\Sigma-1[x-\mu 1])-.33\exp(-[x-\mu 2]T\Sigma-1[x-\mu 2])=0$
$\exp(-\{-xT\Sigma-1\mu 1+xT\Sigma-1\mu 2+\mu T1\Sigma-1\mu 1-\mu T2\Sigma-1\mu 2\})$=>
$-xT\Sigma\hat{}-1\mu 1+xT\Sigma-1\mu 2+\mu T1\Sigma\hat{}-1\mu 1-\mu T2\Sigma-1\mu 2$=>
$xT[3\Sigma\hat{}-1(\mu 2-\mu 1)]+2[\{\mu T/1\ \Sigma\hat{}-1\mu 1-\mu T2\Sigma-1\mu 2\}]$=>
X1 => 0

(d). Make a plot with x1 along horizontal axis, x2 along vertical axis and draw each of the LDA boundaries you found. For each region, write the class label (e.g. "+2") that would be chosen for a new sample that appeared in that region.



(e). How would your result differ if you used QDA? (keeping all the parameters above the same)

Linear and Nonlinear pretty sums up the differences. The results differ consequently, the two often produce comparable results. However, LDA assumes that the observations are drawn from a Gaussian distribution with a common covariance matrix throughout every type of Y, and so can supply some improvements over logistic regression when this assumption approximately holds. Conversely, logistic regression can outperform LDA if these Gaussian assumptions are not met. Both LDA and logistic regression produce linear decision boundaries so when the actual selection boundaries are linear, then the LDA and logistic regression processes will tend to perform well. QDA, on the other-hand, presents a non-linear quadratic decision boundary. Thus, when the choice boundary is relatively non-linear, QDA might also give higher effects.

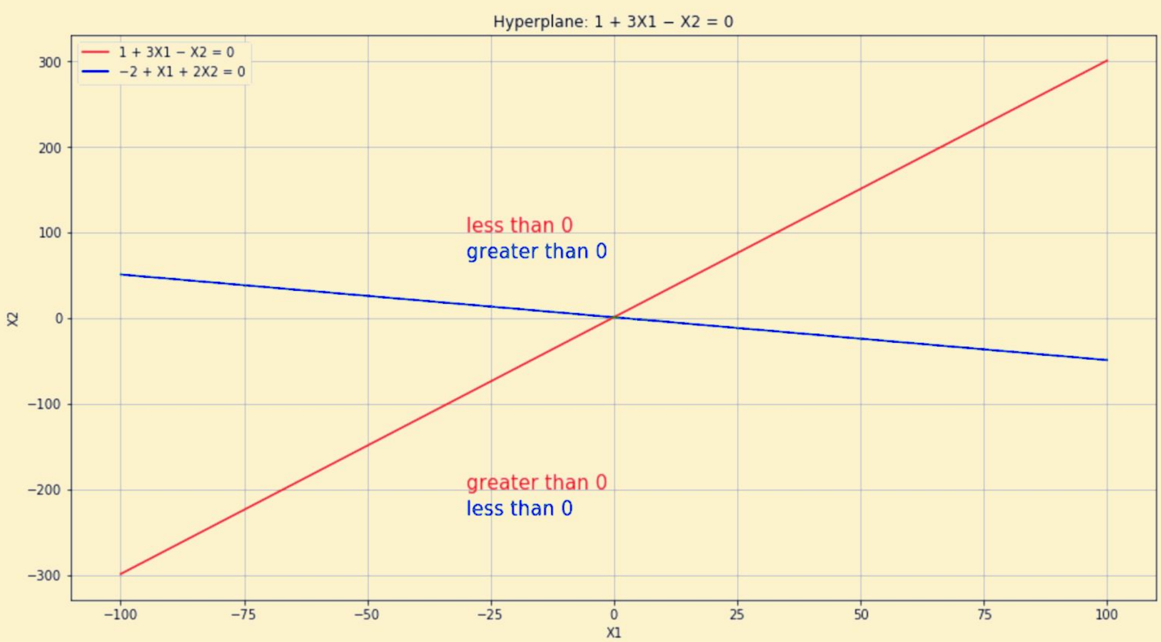
(f). In class, we separately discussed LDA and naive Bayes assumptions. If you wanted to use both for the above problem, explain what would change (if anything).

I wouldn't change anything as the formals and equations are needed for both problems.

3)This problem involves hyperplanes in two dimensions.

(a) Sketch the hyperplane 1 + 3X1 − X2 = 0. Indicate the set of points for which 1 + 3X1 − X2 > 0, as well as the set of points for which 1 + 3X1 − X2 < 0.

(b) On the same plot, sketch the hyperplane −2 + X1 + 2X2 = 0. Indicate the set of points for which −2 + X1 + 2X2 > 0, as well as the set of points for which −2 + X1 + 2X2 < 0.

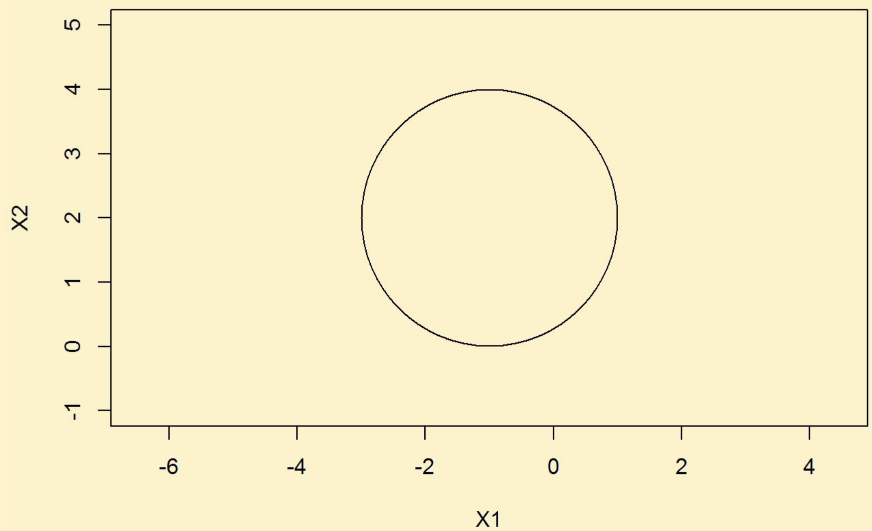


You can see that the lines intersect at zero. The red line steeps up and toward 300 and goes down to -300, while the blue line stays steady.

4) We have seen that in p = 2 dimensions, a linear decision boundary takes the form β0+β1X1+β2X2 = 0. We now investigate a non-linear decision boundary.

(a) Sketch the curve (1 + X1) 2 + (2 − X2) 2 = 4.

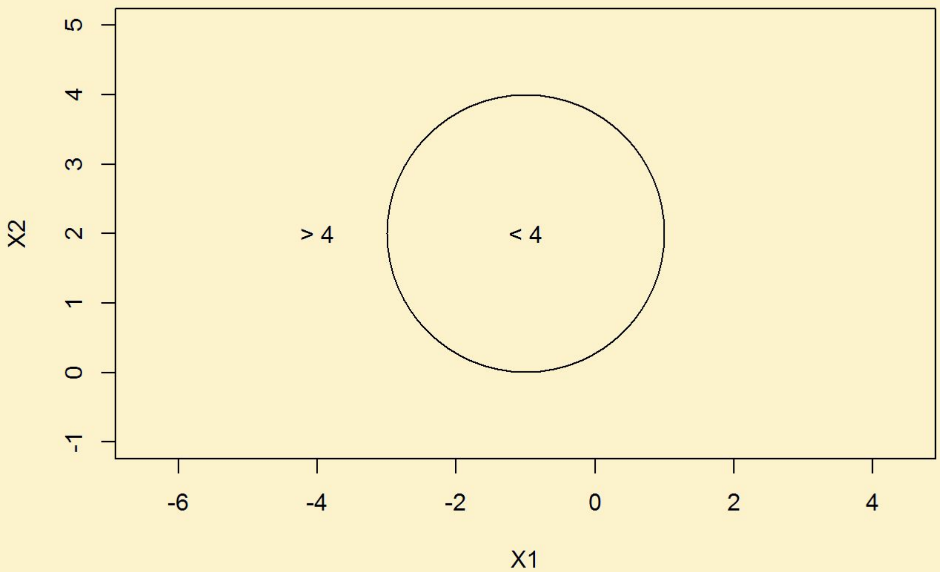
```
plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1", ylab = "X2")
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
```



(b) On your sketch, indicate the set of points for which (1 + X1) 2 + (2 − X2) 2 > 4, as well as the set of points for which (1 + X1) 2 + (2 − X2) 2 ≤ 4.

Python code:

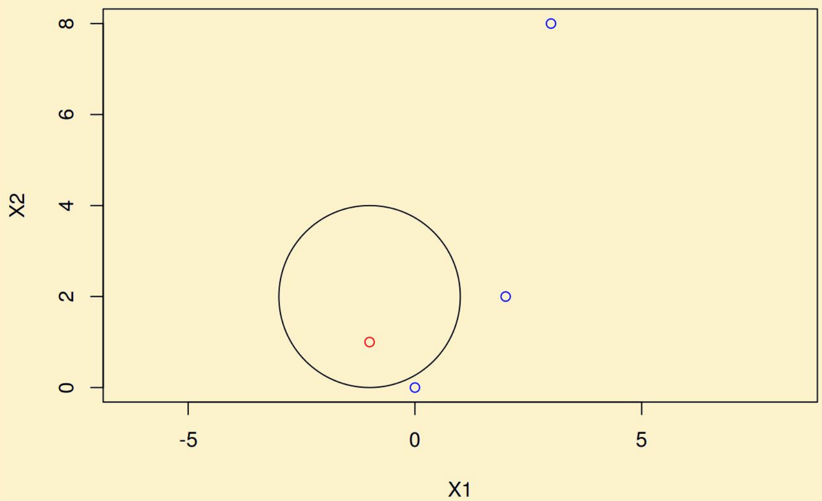
```
plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1", ylab = "X2")
symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
text(c(-1), c(2), "< 4")
text(c(-4), c(2), "> 4")
```



(c) Suppose that a classifier assigns an observation to the blue class if (1 + X1) 2 + (2 − X2) 2 > 4, and to the red class otherwise. To what class is the observation (0, 0) classified? (−1, 1)? (2, 2)? (3, 8)?

```
plot(c(0, -1, 2, 3), c(0, 1, 2, 8), col = c("blue", "red", "blue", "blue"),  type = "p", asp = 1, xlab = "X1",
ylab = "X2") symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
```

It does the trick to supplant X1 and X2 by the directions of the focuses in the condition and to check if the outcome is less or more prominent than 4. For (0,0), we have 5>4 (blue class), for (−1,1), we have 1<4 (red class), for (2,2), we have 9>4 (blue class), for (3,8), we have 52>4 (blue class).



(d) Argue that while the decision boundary in (c) is not linear in terms of X1 and X2, it is linear in terms of X1, X 2/1 , X2, and X2/2 .

It is apparent seeing that we may extend the equation of the decision boundary:

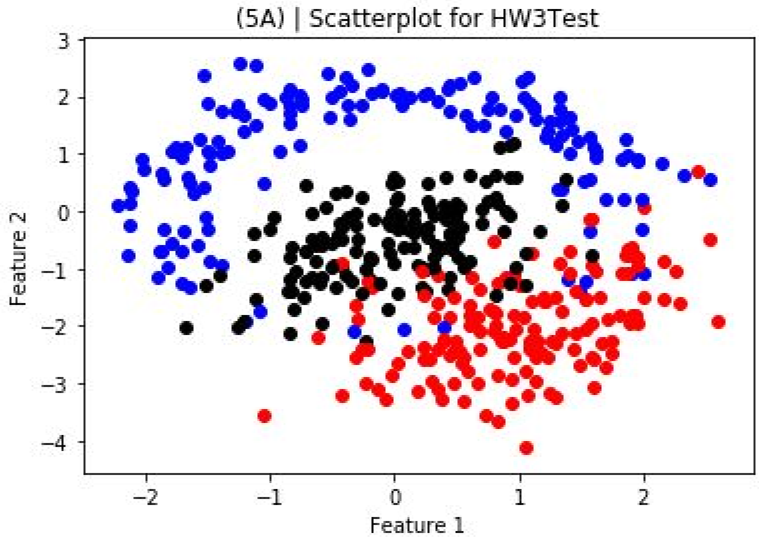
$$(1 + X_1)^2 + (2 - X_2)^2 = 4$$

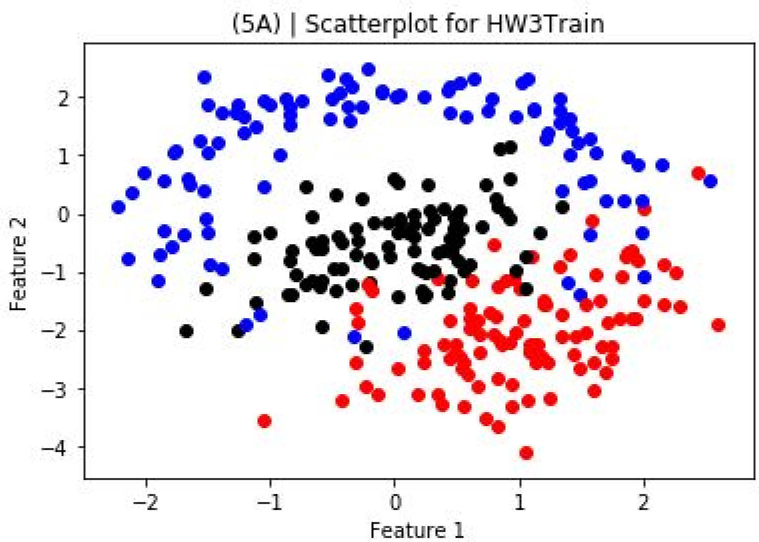
by

$$X_1^2 + X_2^2 + 2X_1 - 4X_2 + 1 = 0$$

which is linear in terms of X1, X 2/1, X2 and X 2/2

5) Programming Section:

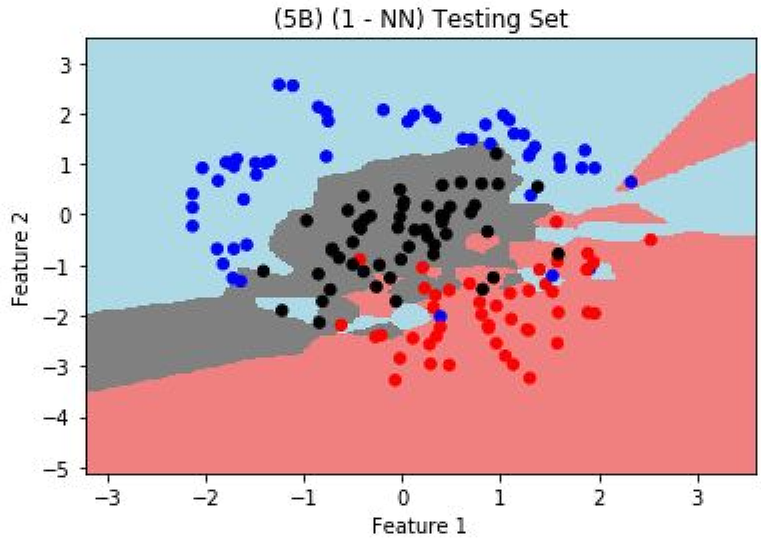


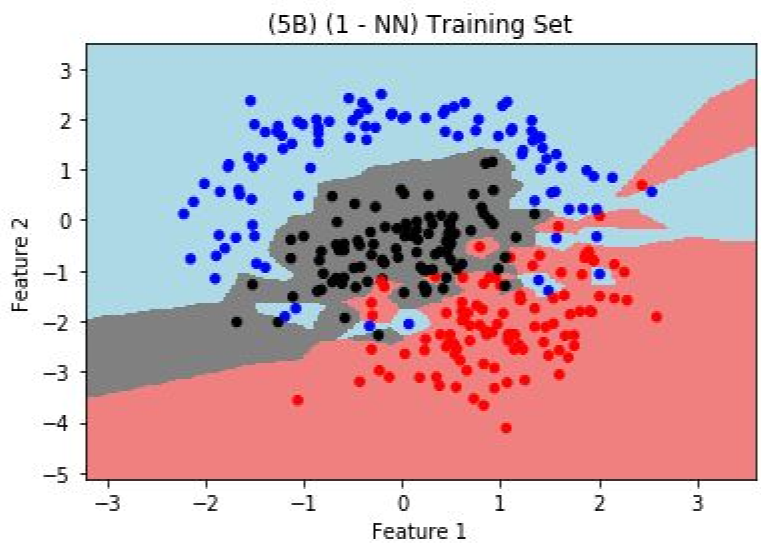


(5B) | K Neighbors Classifier fo k = 1

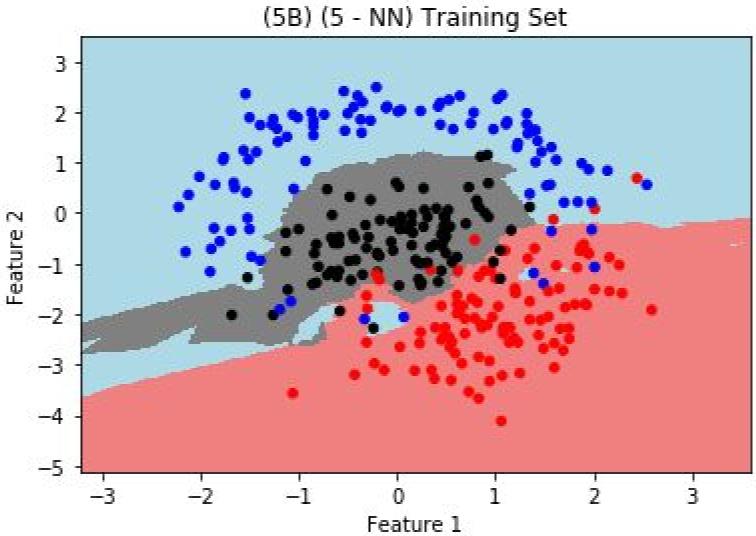
Training Accuracy is 1.00

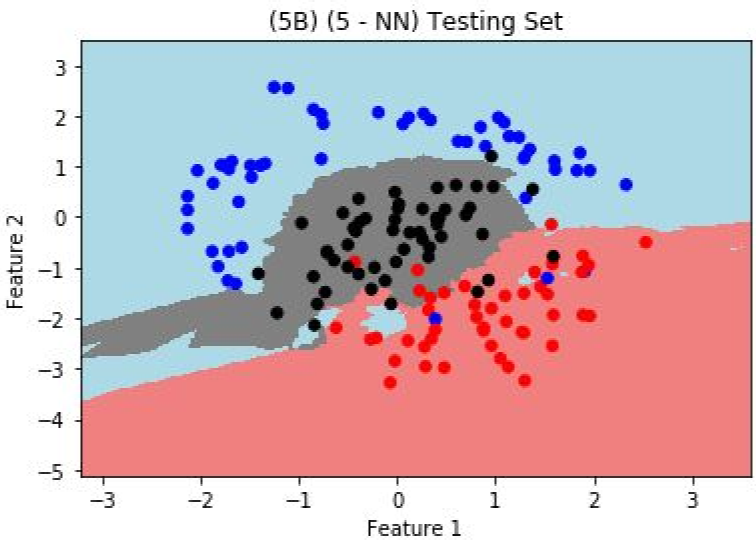
Testing Accuracy is 0.81





(5B) | K Neighbors Classifier fo k = 5
Training Accuracy is 0.93
Testing Accuracy is 0.87

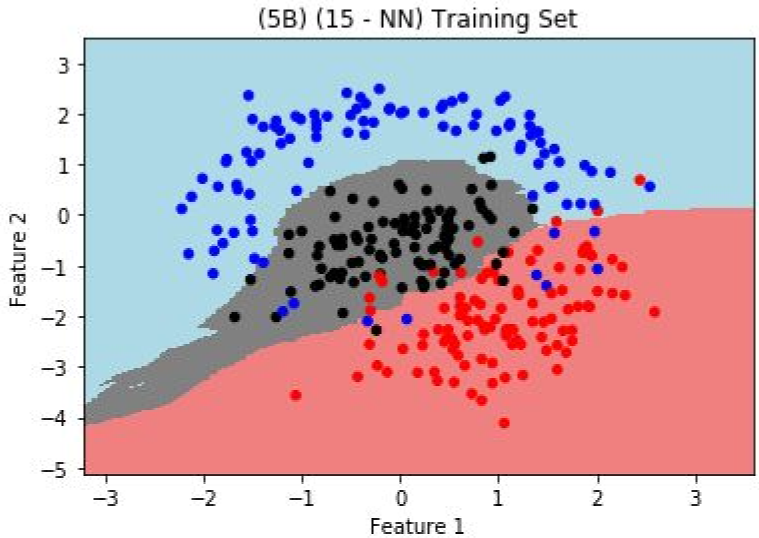


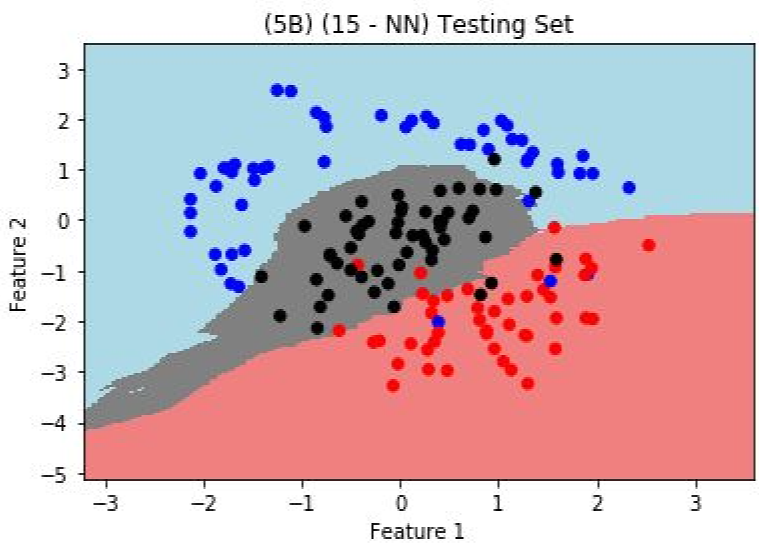


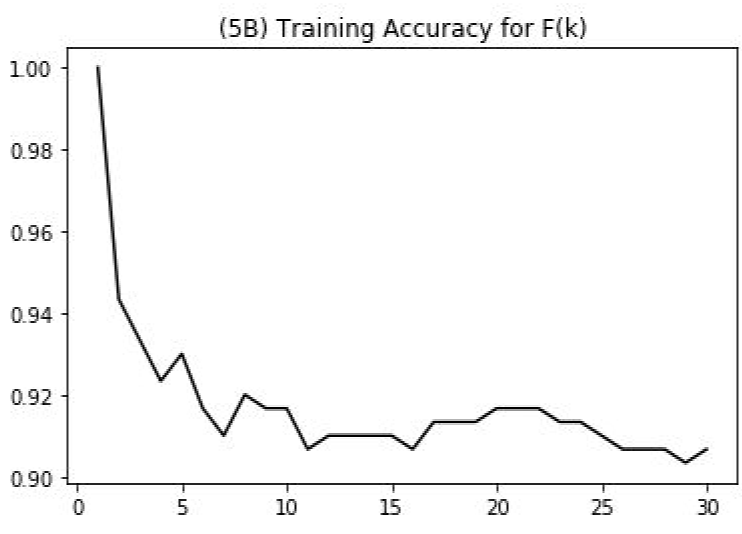
(5B) | K Neighbors Classifier fo k = 15

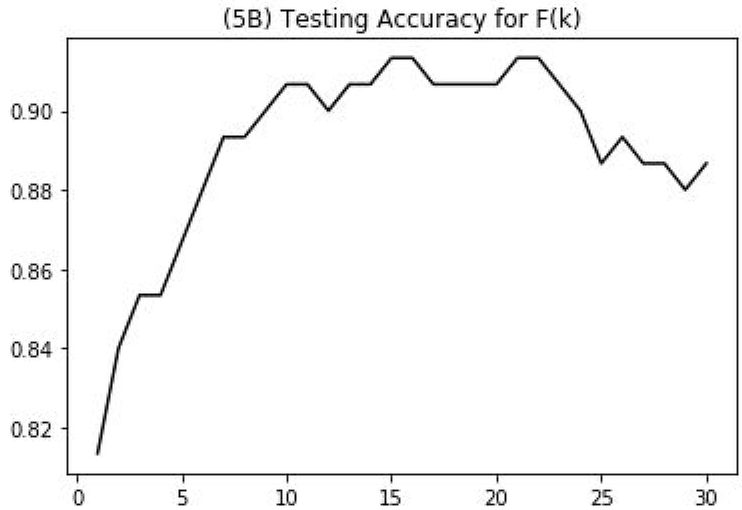
Training Accuracy is 0.91

Testing Accuracy is 0.91

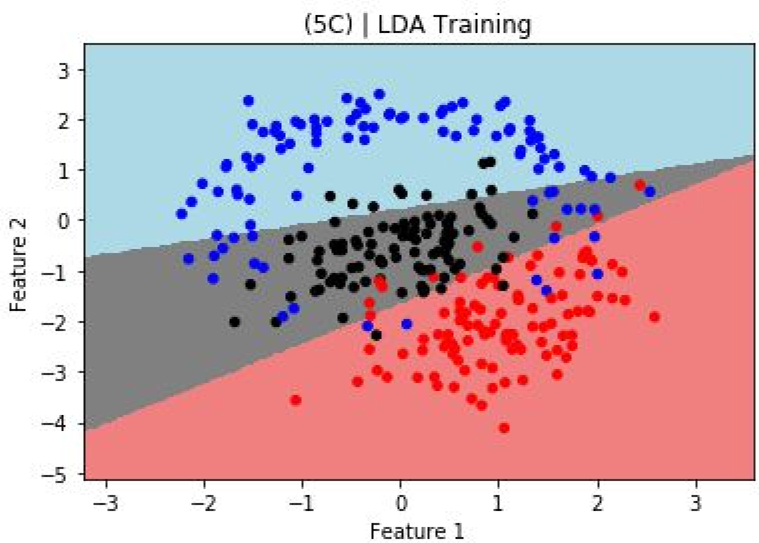

(5B) (15 - NN) Training Set
Feature 2
Feature 1
3
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1
2
3


(5B) (15 - NN) Testing Set
Feature 2
Feature 1
3
2
1
0
−1
−2
−3
−4
−5
−3
−2
−1
0
1
2
3

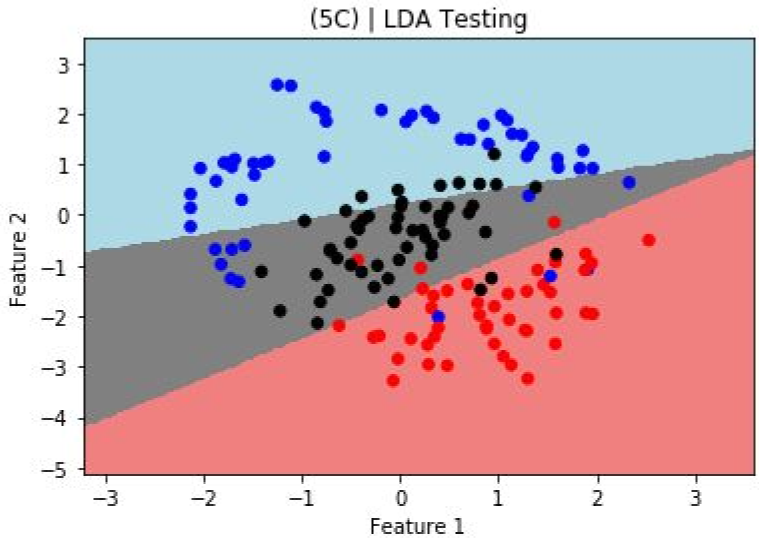

(5B) Training Accuracy for F(k)
1.00
0.98
0.96
0.94
0.92
0.90
0
5
10
15
20
25
30



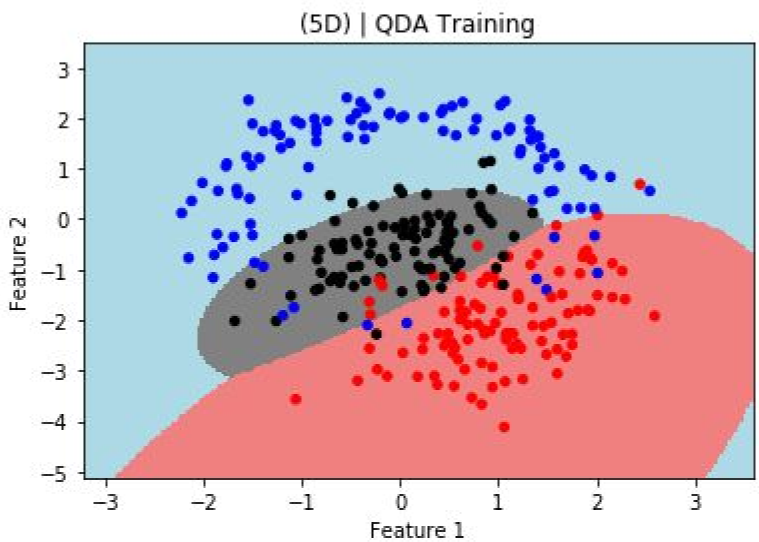
(5C) | LDA - Score - Training = 0.8333333333333334



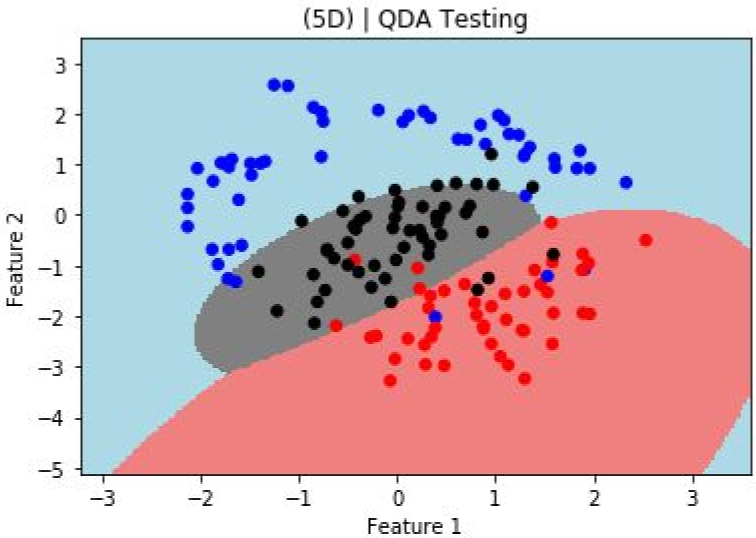
(5C) | LDA - Score - Testing = 0.8133333333333334



(5D) | QDA - Score - Training = 0.8966666666666666



(5D) | QDA - Score - Testing = 0.8666666666666667

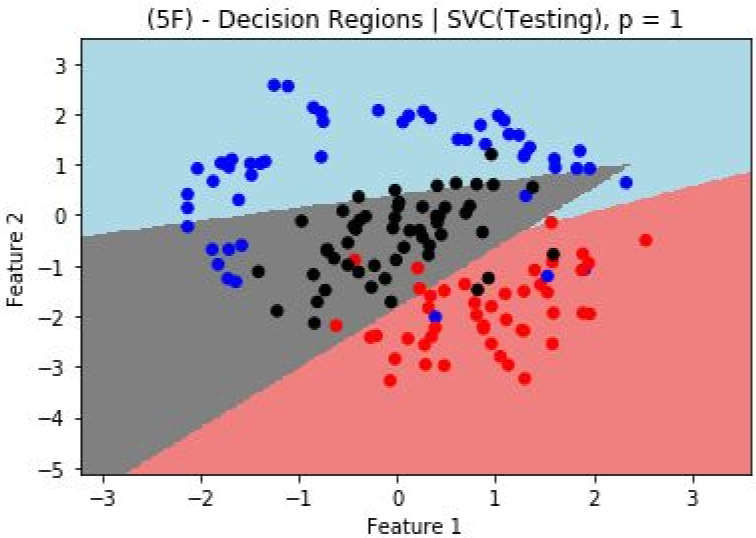


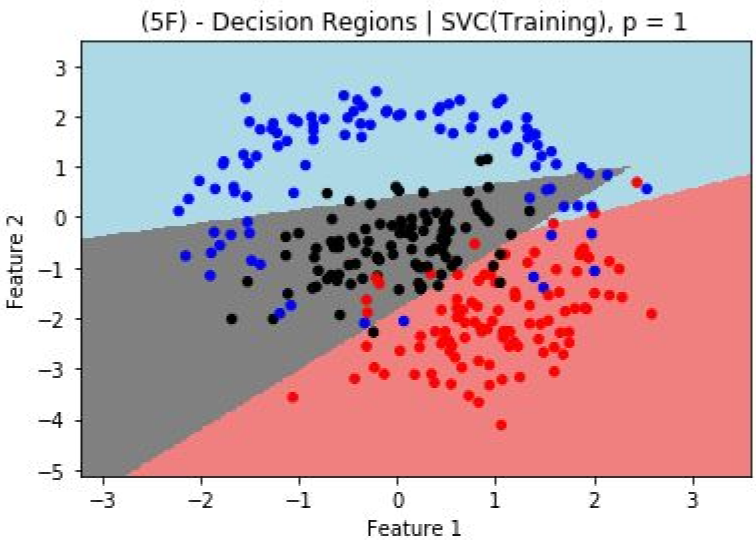
5E) Between LDA & QDA it's important to understand that LDA is an awful a lot much less bendy classifier than QDA, and so has substantially decreased variance. This can doubtlessly lead to improved prediction performance. But there is a trade-off: if LDA's assumption that the predictor variable shares a common variance throughout every Y response classification is badly off, then LDA can go through from high bias. Roughly speaking, LDA tends to be a higher wager than QDA if there are tremendously few coaching observations and so reducing variance is crucial. In contrast, QDA is advocated if the training set is very large, so that the variance of the classifier is now not a predominant concern, or if the assumption of a frequent covariance matrix is in reality untenable.

Method	Pros	Cons
KNN	notable classification results no (re)training phase distance metrics error probability bounded	time consuming classification time memory utilization finding optimal k
LDA	linear decision boundary fast classification easy to implement	Gaussian assumptions training time complex matrix ops
QDA	quadratic decision boundary fast classification classification more accurate outperforms KNN and LDA	Gaussian assumptions training time complex matrix ops

(5F) For 'p' = 1
Best 'C value' = 5.623413251903491
Training 'Accuracy' = 0.8533333333333334

Testing 'Accuracy' = 0.8466666666666667



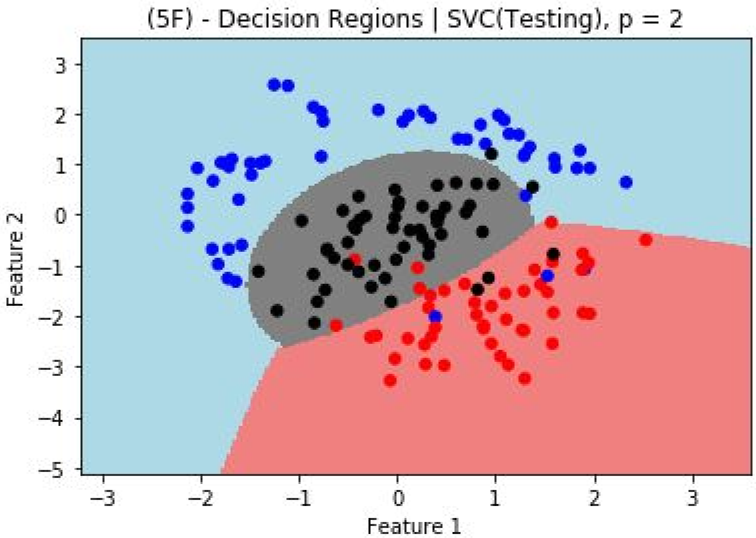


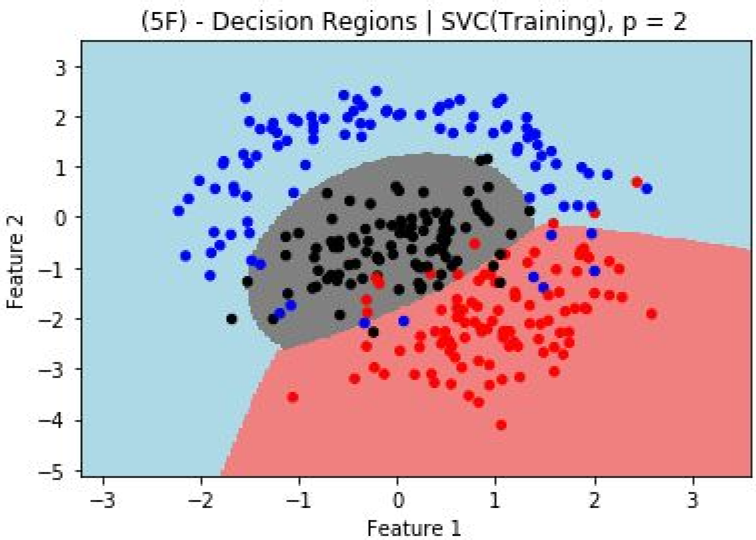
(5F) For 'p' = 2

Best 'C value' = 0.1

Training 'Accuracy' = 0.9066666666666666

Testing 'Accuracy' = 0.92



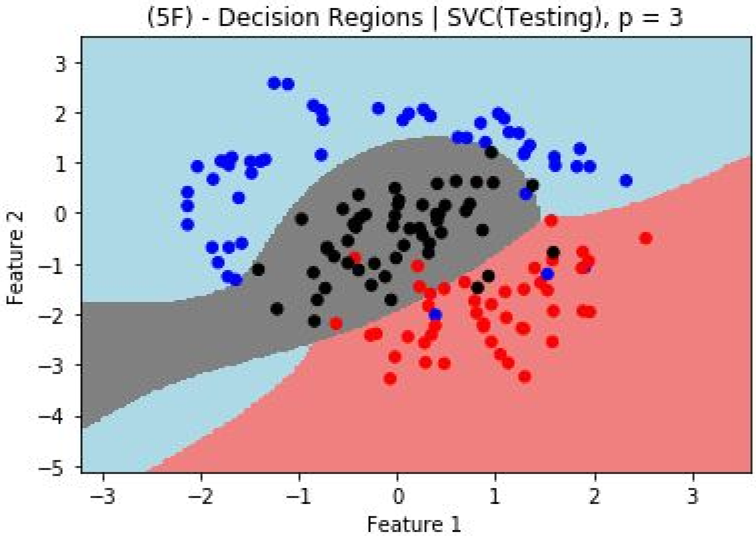


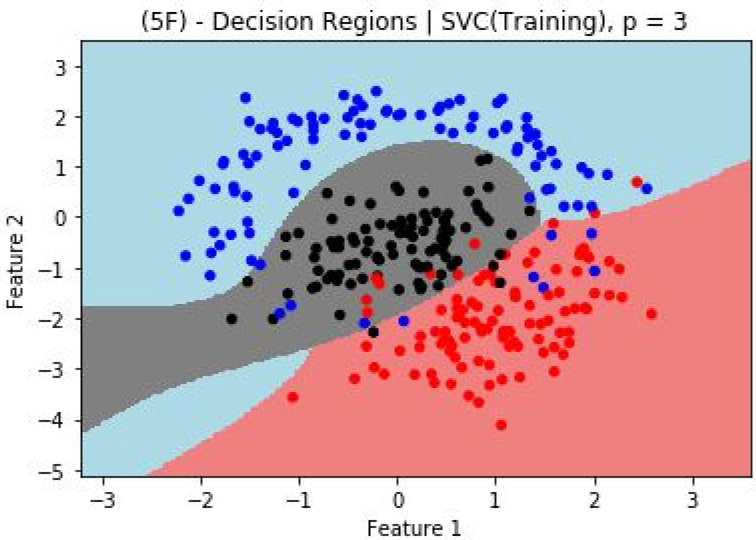
(5F) For 'p' = 3

Best 'C value' = 0.31622776601683794

Training 'Accuracy' = 0.9233333333333333

Testing 'Accuracy' = 0.92



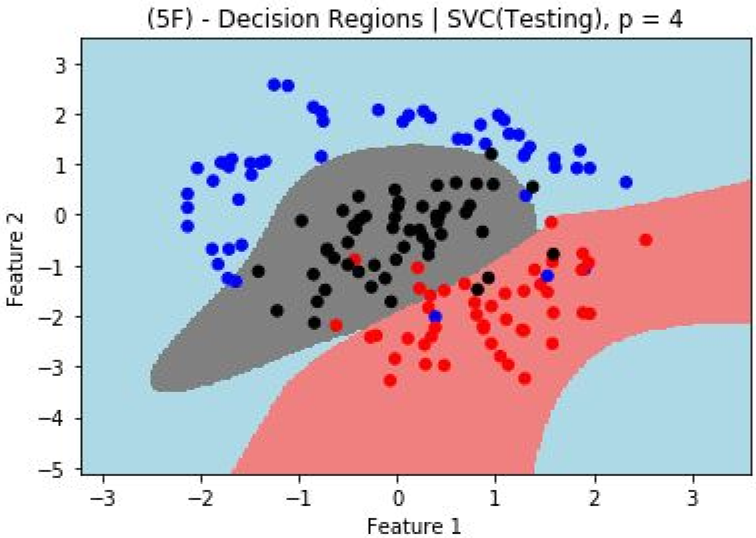


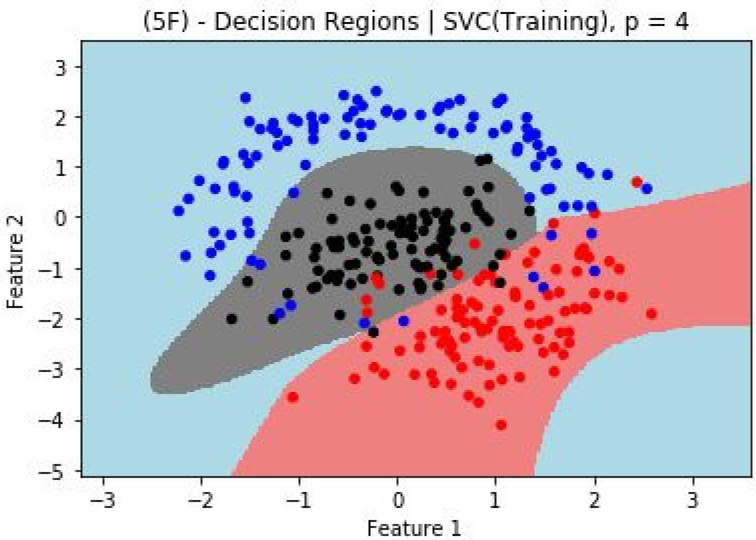
(5F) For 'p' = 4

Best 'C value' = 0.01

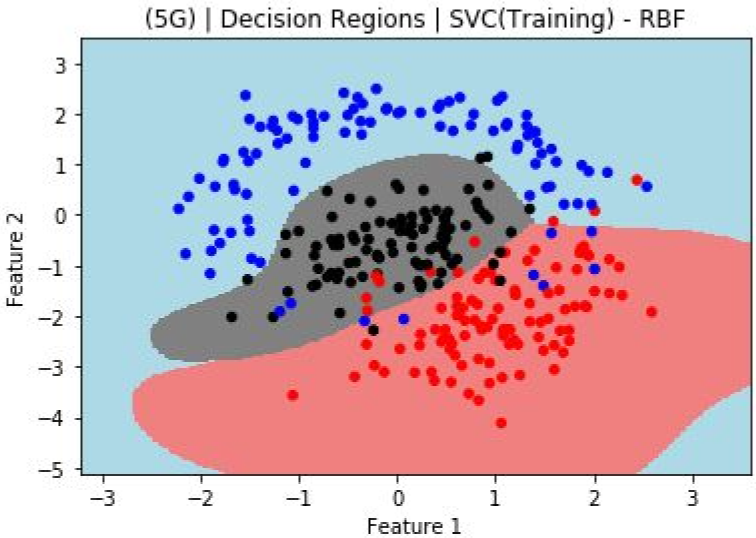
Training 'Accuracy' = 0.9133333333333333

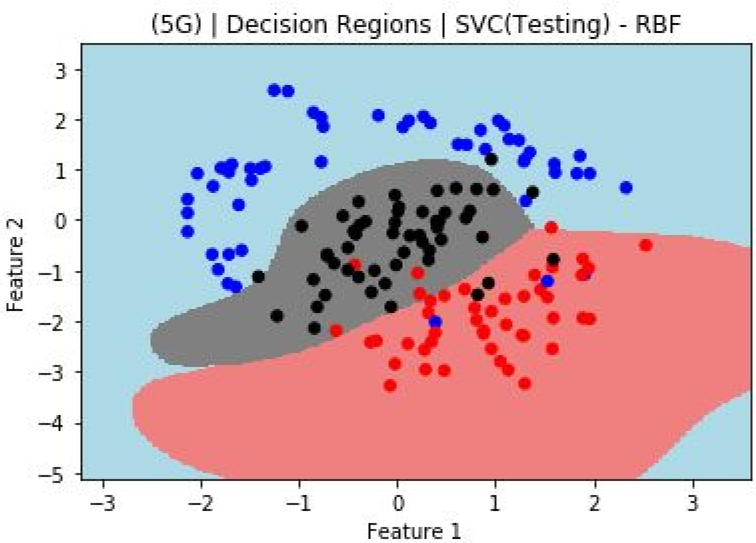
Testing 'Accuracy' = 0.9066666666666666





(5G) | SVM with 'Radial Bias Function' ->
[1] Best 'C Value' = 0.9266666666666666
[2] Best 'Gamma Value' = 0.91
[3] Training 'Accuracy' = 0.91
[4] Testing 'Accuracy' = 0.9266666666666666





H) As KNN works better than SVM, it suggests that your information set is no longer without problems separable the use of the decision planes that you have let SVM use; i.e. the simple SVM uses linear hyperplanes to separate classes, and if you provide a exceptional kernel, then that will alternate the form of the choice manifold that can be used. two KNN can generate a quite convoluted decision boundary as it is driven by using the uncooked education statistics itself. For example, suppose of how a Voronoi format can separate a couple of areas with a non-convex boundary made up of piecewise linear hyperplanes; KNN in the end behaves in a comparable way. SVM makes use of an enormously restricted parametric approximation of the decision boundary, which is an outstanding trade-off for classification performance towards statistics storage space/ processing speed.
If KNN can furnish appropriate results, then it suggests that your instructions are pretty separable; if KNN failed, then it would point out that the metric vector you have chosen does no longer produce separable classes.

I) Looking at the previous data generated along with the boundaries and graphics generated. I think it's clear to see in some instances that SVM performed a little better in some areas than kNN.