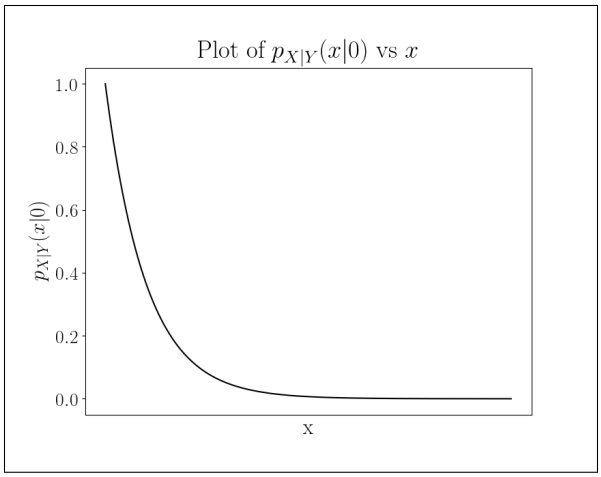
Problem Set 1

NAME: Harsh Bhate **GTID: 903424029**

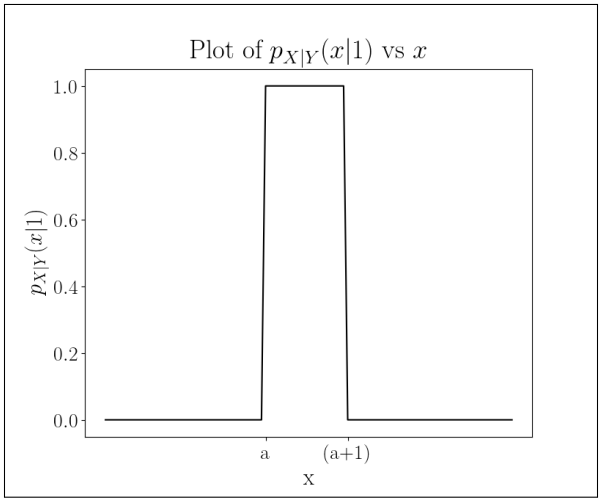
Problem	Points
Problem 1	
Problem 2	
Problem 3	
Problem 4	
Total	

Problem 1

(a).



(a) Plot of $p_{X|Y}(x|0)$



(b) Plot of $p_{X|Y}(x|1)$

Figure 1: Plots of $p_{X|Y}(x|0)$ and $p_{X|Y}(x|1)$

(b). The solution to an Optimal Bayes Classifier ($h_k^B(x)$) is:

$$\begin{aligned} h_k^B(x) &= \underset{k}{\text{argmax}}\ p_{y|x}(k|x) \\ &= \underset{k}{\text{argmax}}\ \frac{p_{x|y}(x|k)p_y(k)}{p_x(x)} \\ &= \underset{k}{\text{argmax}}\ \frac{1}{2}\frac{p_{x|y}(x|k)}{p_x(x)} \\ &= \underset{k}{\text{argmax}}\ \frac{1}{2}\ p_{x|y}(x|k) \end{aligned}$$

Upon observing the limits providing by the distribution of $p_{x|y}$, we conclude:

$$h_k^B(x) = \begin{cases} 1 & \text{if } x \in [a, (a+1)] \\ 0 & \text{otherwise} \end{cases}$$

(c). The Risk is defined as:

$$\begin{aligned} R(h_k^B(x)) &= \mathbb{E}(\mathbb{1}\{h_k^B(x) \neq y\}) \\ &= \mathbb{E}_x\mathbb{E}_{y|x}(\mathbb{1}\{h_k^B(x) \neq y\}|x) \\ &= \mathbb{E}_x(1 - \mathbb{E}_{y|x}(\mathbb{1}\{h_k^B(x) = y\}|x)) \\ &= 1 - \mathbb{E}_x(\mathbb{E}_{y|x}(\mathbb{1}\{h_k^B(x) = y\}|x)) \end{aligned}$$

Let $\alpha = \mathbb{E}_{y|x}(\mathbb{1}\{h_k^B(x) = y\}|x)$. Thus,

$$R(h_k^B(x)) = 1 - \mathbb{E}_x(\alpha)$$

Let us now simplify α:

$$\begin{aligned} \alpha &= \mathbb{E}_{y|x}(\mathbb{1}\{h_k^B(x) = y\}|x) \\ &= p_{y|x}(h_k^B(x) = y) \\ &= \sum_y p_{y|x}(y|x)\mathbb{1}\{h_k^B(x) = y\} \\ &= \sum_y p_{y|x}(y|x)\mathbb{1}\{\underset{k}{\text{argmax}}\ p_{y|x}(k|x) = y\} \\ &= p_{y|x}\big(\underset{k}{\text{argmax}}\ p_{y|x}(k|x)|x\}\big) \\ &= \max_h\ p_{y|x}(k|x) \end{aligned}$$

Plugging the simplified α into the risk equation, we get:

$$\begin{aligned} R(h_k^B(x)) &= 1 - \mathbb{E}_x(\max_h p_{y|x}(k|x)) \\ &= 1 - \int \max_h p_{x|y}(x|k)p_y(k)dx \\ &= 1 - \frac{1}{2}\int \max_h p_{x|y}(x|k)dx \end{aligned}$$

Let's simplify

$$\begin{aligned} \int \max_h p_{x|y}(x|k)dx &= \int_0^a e^{-x}dx + \int_a^{a+1} dx + \int_{a+1}^{\infty} e^{-x}dx \\ &= -e^{-x}|_0^a + x|_a^{a+1} + -e^{-x}|_{a+1}^{\infty} \\ &= (-e^{-a}+1) + (a+1-a) + (0+e^{-(a+1)}) \\ &= 2 - e^{-a} + e^{-a}e^1 \\ &= 2 + e^{-a}(e^1 - 1) \end{aligned}$$

Thus, the risk is:

$$\begin{aligned} R(h_k^B(x)) &= 1 - \frac{1}{2}\int \max_h p_{x|y}(x|k)dx \\ &= 1 - \frac{1}{2}\big(2 + e^{-a}(e^1 - 1)\big) \\ &= \frac{e^{-a}}{2}(1 - e^1) \end{aligned}$$

(d). The Bayes classifier with estimate $\hat{a}$ is:

$$h^B_{k\,\hat{a}}(x) = \begin{cases} 1 & \text{if } x \in [\hat{a}, (\hat{a}+1)] \\ 0 & \text{otherwise} \end{cases}$$

The risk for this new classifier is:

$$\begin{aligned} R(h^B_{k\,\hat{a}}(x)) &= \mathbb{E}(\mathbb{1}\{h^B_{k\,\hat{a}}(x) \neq y\}) \\ &= P(h^B_{k\,\hat{a}}(x) \neq y) \\ &= P(Y=0)P_{x|0}(h^B_{\hat{a}}(x) \neq 0) + P(Y=1)P_{x|1}(h^B_{\hat{a}}(x) \neq 1) \quad \text{[Law of Total Probability]} \\ &= \frac{1}{2}\left(P_{x|0}(h^B_{\hat{a}}(x) \neq 0) + P_{x|0}(h^B_{\hat{a}}(x) \neq 1)\right) \end{aligned}$$

Let us now evaluate $P_{x|0}(h^B_{\hat{a}}(x) \neq 0)$ and $P_{x|0}(h^B_{\hat{a}}(x) \neq 1)$ indiviually and then plug in the equation above.

$$P_{x|0}(h^B_{\hat{a}}(x) \neq 0) = \int_0^\infty e^{-x}\mathbb{1}\{(h^B_{\hat{a}}(x) \neq 0)\}dx$$

From the classifier definition, the classifier is not 0 when $x \in [\hat{a}, (\hat{a}+1)]$. Thus,

$$\begin{aligned} P_{x|0}(h^B_{\hat{a}}(x) \neq 0) &= \int_{\hat{a}}^{\hat{a}+1} e^{-x}dx \\ &= -e^{-x}|_{\hat{a}}^{\hat{a}+1} \\ &= e^{-\hat{a}} - e^{-(\hat{a}+1)} \\ &= e^{-\hat{a}}(1-e^{-1}) \end{aligned}$$

Similarly,

$$P_{x|1}(h^B_{\hat{a}}(x) \neq 1) = \int_a^{(a+1)} \mathbb{1}\{(h^B_{\hat{a}}(x) \neq 1)\}dx$$

From the classifier definition, the classifier is not 1 when $x \notin [\hat{a}, (\hat{a}+1)]$. We now split the problem in two case:

$$\begin{aligned} P_{x|1}(h^B_{\hat{a}}(x) \neq 1) &= \begin{cases} \int_{a+1}^{\hat{a}+1} dx & \text{if } a < \hat{a} \\ \int_{\hat{a}+1}^{a+1} dx & \text{if } \hat{a} \leq a \end{cases} \\ &= \begin{cases} x|_{a+1}^{\hat{a}+1} & \text{if } a < \hat{a} \\ x|_{\hat{a}+1}^{a+1} & \text{if } \hat{a} \leq a \end{cases} \\ &= \begin{cases} \hat{a} - a & \text{if } a < \hat{a} \\ a - \hat{a} & \text{if } \hat{a} \leq a \end{cases} \\ &= |\hat{a} - a| \end{aligned}$$

Thus, the risk of the new Classifier is:

$$\boxed{R(h^B_{\hat{a}}(x)) = \begin{cases} \frac{1}{2}\left(e^{-\hat{a}}(1-e^{-1}) + |\hat{a}-a|\right) & \text{if } |\hat{a}-a| \leq 1 \\ \frac{1}{2}\left(e^{-\hat{a}}(1-e^{-1}) + 1\right) & \text{if } |\hat{a}-a| > 1 \end{cases}}$$

(e). The Risk of the classifier given the error is:

$$\begin{aligned} R(h^B_{\hat{a}}(x)) &= \mathbb{E}(\mathbb{1}\{h^B_{\hat{a}(x)} \neq y\}) \\ &= \mathbb{P}(h^B_{\hat{a}}(x) \neq y \;||\hat{a}-a| \geq \epsilon)\mathbb{P}(|\hat{a}-a| \geq \epsilon) + \mathbb{P}(h^B_{\hat{a}}(x) \neq y \;||\hat{a}-a| < \epsilon)\mathbb{P}(|\hat{a}-a| < \epsilon) \\ &\leq \max\, \mathbb{P}(h^B_{\hat{a}}(x) \neq y \;||\hat{a}-a| \geq \epsilon)\delta + \max\, \mathbb{P}(h^B_{\hat{a}}(x) \neq y \;||\hat{a}-a| < \epsilon)(1-\delta) \end{aligned}$$

Let us now evaluate the Right Hand Side separately,

$$\max\, \mathbb{P}(h^B_{\hat{a}}(x) \neq y \;||\hat{a}-a| \geq \epsilon) = \frac{1}{2}(e^{-\hat{a}}(1-e^{-1}) + 1)$$

Also,

$$\max\, \mathbb{P}(h^B_{\hat{a}}(x) \neq y \;||\hat{a}-a| < \epsilon) = \frac{1}{2}(e^{-\hat{a}}(1-e^{-1}) + \epsilon)$$

Thus, the Risk is bounded by:

$$R(h^B_{\hat{a}}(x)) \leq \frac{1}{2}\left[(e^{-\hat{a}}(1-e^{-1}) + 1)\delta + (1-\delta)(e^{-\hat{a}}(1-e^{-1}) + \epsilon)\right]$$

Simplifying, we get,

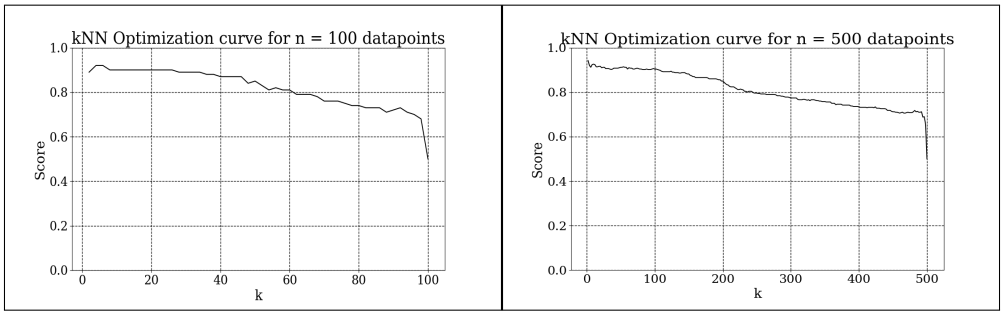
$$\boxed{R(h^B_{\hat{a}}(x)) \leq \frac{1}{2}\left[e^{-\hat{a}}(1-e^{-1}) + \epsilon(1-\delta) + \delta\right]}$$

Problem 2

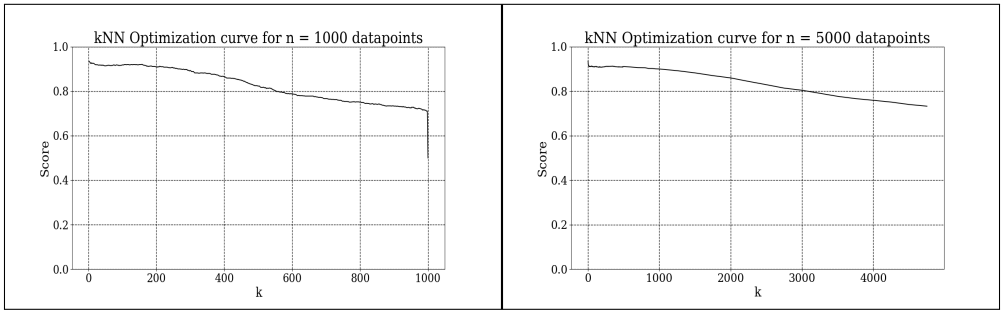
To determine the optimum value of k for a given number of data point, the score (or net accuracy) of the model on the training set was used. While the training set accuracy isn't the best indicator of the performance characteristic of a classifier, it serves as a good quantifiable measure. Also, the following values of k were ignored in the consideration of optimal k:

- $\mathbf{k = 1}$. A 1-NN classifier is an ideal classifier and thus does not serve practical. This is so because the classifier uses only the data point and not it's neighbor. This inherently makes the classifier unfeasible.
- $\mathbf{k \in Even}$. An even kNN classifier would lead to a decision paralysis in case of a split vote resulting in poor performance.

Using the built-in *score* method in python, accuracy figures for different values of k were plotted and the highest score yielded the best result.



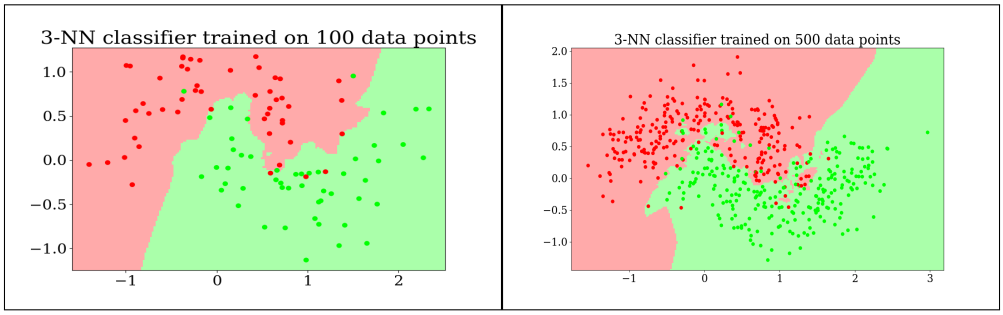
(a) Optimization Curve for 100 datapoints (b) Optimization Curve for 500 datapoints



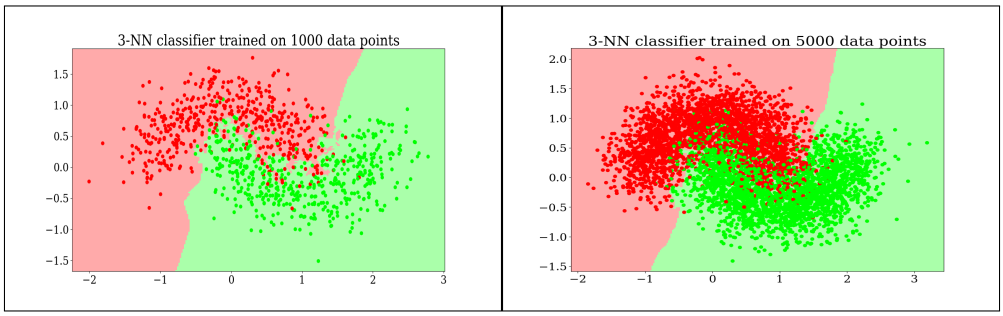
(c) Optimization Curve for 100 datapoints (d) Optimization Curve for 5000 datapoints

Figure 2: Optimization Curves for different number of datapoints

The value of k corresponding to the largest value of score was chosen as the optimal k. The results are as follows:



(a) Optimal kNN for 100 datapoints (b) Optimal kNN for 500 datapoints



(c) Optimal kNN for 1000 datapoints (d) Optimal kNN for 5000 datapoints

Figure 3: Decision plots for Optimal K based on Score Heuristics

The Optimal Value of k based on the score heuristics is:

N (Datapoints)	Optimal k	Score
100	3	0.92
500	3	0.942
1000	3	0.935
5000	3	0.9348

Table 1: Optimal k values based on Scoring heuristics for different Datapoints

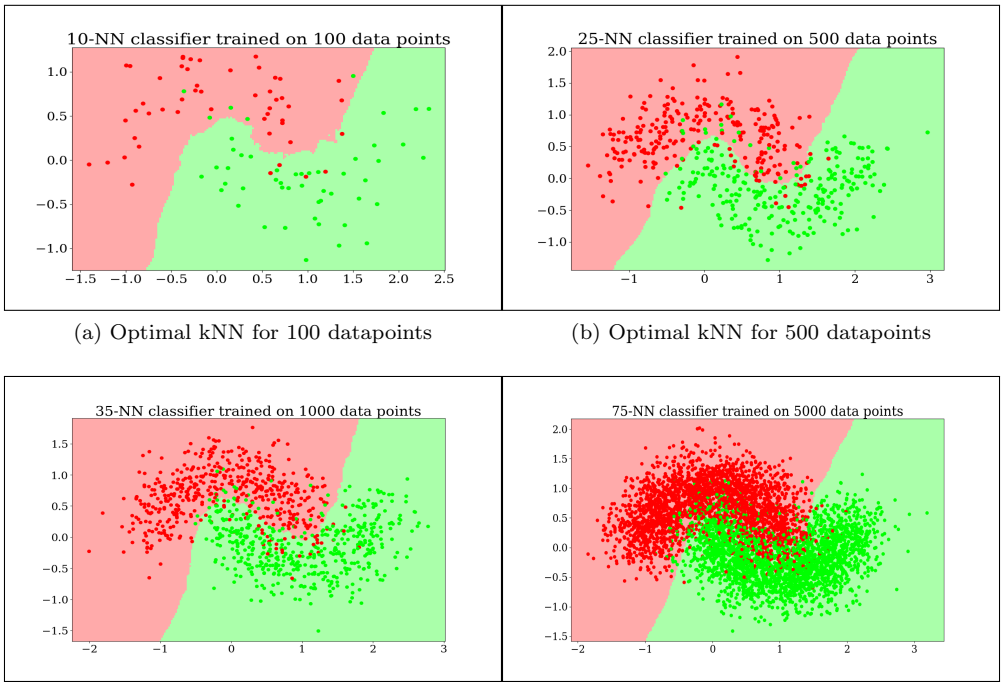
While the heuristics of score tend to prefer the lower value of the k. This is so because the lower the value of k, the closer it is to overfitting as the dataset becomes larger. The large scale of relations introduced by higher value of k are ignored. This yields the classifier to be highly prone to overfitting and results in a loss of generalization. This can be observed in the decision plot above.

From a visual perspective and from a generalization standpoint, the following values of k seem more optimal:

N (Datapoints)	Optimal k	Score
100	10	0.9
500	25	0.908
1000	35	0.918
5000	75	0.9094

Table 2: Optimal k values based on Visual heuristics for different Datapoints

The corresponding results are:



(a) Optimal kNN for 100 datapoints

(b) Optimal kNN for 500 datapoints

(c) Optimal kNN for 1000 datapoints

(d) Optimal kNN for 5000 datapoints

Figure 4: Decision plots for Optimal K based on Score Heuristics

Problem 3

(a). Since the features are conditionally independent and Bernoulli distributed,

$$\mathbb{P}(\mathbf{X}, \mu) = \prod_{i=0}^{1} \mu^{x_i}(1-\mu)^{1-x_i}$$

Taking log on both sides, we get,

$$log(\mathbb{P}(\mathbf{X}, \mu)) = \sum_{i=0}^{1} \Big[x_i log(\mu) + (1-x_i) log(1-\mu) \Big]$$

Since log is a monotonic function,

$$\underset{\mu}{\text{argmax}}\ \mathbb{P}(\mathbf{X}, \mu) = \underset{\mu}{\text{argmax}}\ log(\mathbb{P}(\mathbf{X}, \mu))$$

Thus, to Maximise we find the partial derivative with respect to μ.

$$\begin{aligned}\frac{\partial log(\mathbb{P}(\mathbf{X}, \mu))}{\partial \mu} &= \underset{\mu}{\text{argmax}} \sum_{i=0}^{1} \frac{\partial}{\partial \mu} \Big[x_i log(\mu) + (1-x_i) log(1-\mu) \Big] \\ &= \frac{1}{\mu} \sum_{i=0}^{1} x_i - \frac{1}{1-\mu} \sum_{i=0}^{1} (1-x_i)\end{aligned}$$

Equating to zero, we get,

$$\frac{1}{\mu} \sum_{i=0}^{1} x_i = \frac{1}{1-\mu} \sum_{i=0}^{1} (1-x_i)$$

Thus,

$$\boxed{\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i = \frac{N_i}{N}}$$

Plugging the values using the equation above, we can conclude,

$$\boxed{\hat{\mu}_{1,0} = \hat{\mu}_{1,1} = \frac{N}{N} = 1}$$

(b). The output of the Naïve Bayes Classifier is based on:

$$\mathbb{P}(y=1|X) = \frac{\mathbb{P}(y=1)\mathbb{P}_{x|Y}(X|y=1)}{\mathbb{P}_x(X)}$$

Note that $\mathbb{P}_{x|y}(x|1) = 0$. Thus, because of the Binary nature of features, $\mathbb{P}_{x|y}(0|1) = 0$. Also,

$$\mathbb{P}_x = \mathbb{P}_{x|y}(x|0)\mathbb{P}(Y=0) + \mathbb{P}_{x|y}(x|1)\mathbb{P}(Y=1) = 0$$

Therefore,

$$\mathbb{P}(y=1|X) = \frac{\mathbb{P}(y=1) \cdot 0}{\mathbb{P}_x(x)} = \frac{0}{0}$$

If a feature is continually recurring throughout the sample, the classifier learns to ignore that feature. For example, while comparing humans and animals the feature eyes is ignored as both human and monkeys have eyes. This results in the classifier overfitting on eye feature.

(c). The MAP estimation is defined as:

$$\begin{aligned}\underset{\mu}{\text{argmax}}\ \mathbb{P}(\mu|X) &= \underset{\mu}{\text{argmax}}\ log(\mathbb{P}(\mu|X)) \\ &= \underset{\mu}{\text{argmax}}\ log(\prod_{x_i}\mathbb{P}(x_i|\mu)\mathbb{P}(\mu)) \\ &= \underset{\mu}{\text{argmax}}\ \sum_{x_i} log(\mathbb{P}(x_i|\mu)) + log(\mathbb{P}(\mu)) \\ &= \mathcal{L}\end{aligned}$$

To find the maximum, we differentiate with respect to μ. Thus,

$$\frac{\partial\mathcal{L}}{\partial\mu} = \underset{\mu}{\text{argmax}}\ \sum_{x_i}\frac{\partial}{\partial\mu}log(\mathbb{P}(x_i|\mu)) + \frac{\partial}{\partial\mu}log(\mathbb{P}(\mu))$$

Independently evaluating each partial derivative, we get,

$$\begin{aligned}\frac{\partial log(\mathbb{P}(\mu))}{\partial\mu} &= \frac{\partial log(Beta(\beta_0,\beta_1))}{\partial\mu} \\ &= \frac{\beta_0-1}{\mu} - \frac{\beta_1-1}{1-\mu}\end{aligned}$$

And

$$\sum\frac{\partial log(\mathbb{P}(x_i|\mu))}{\partial\mu} = \frac{1}{\mu}\sum x_i - \frac{1}{1-\mu}\sum(1-x_i)$$

Equating $\frac{\partial\mathcal{L}}{\partial\mu} = 0$, we get,

$$\frac{1}{\mu}\sum x_i - \frac{1}{1-\mu}\sum(1-x_i) + \frac{\beta_0-1}{\mu} - \frac{\beta_1-1}{1-\mu} = 0 \tag{1}$$

Note that $\sum x_i = N_i$. Simplifying, we get,

$$\boxed{\mu_{j,k}^{\text{MAP}} = \frac{N_{j,k}+\beta_0-1}{N_K+\beta_0+\beta_1-2}}$$

Problem 4

(a).

kNN Classifier

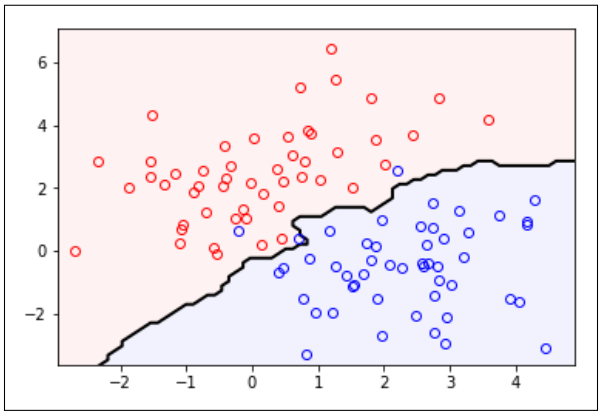


Figure 5: Prediction of kNN *(k = 2)* Classifier

Generalization Capability of kNN

Being a Lazy learning algorithm, kNN does not generalize well enough due to minimal learning steps. As evident in this example, the 2 nearest neighbor classifier overfits the data and also leads to decision paralysis at the edge cases due to lack of majority. This can be further substantiated by the decision boundary being coarse and fitted to the data.

(b).

kNN Classifier

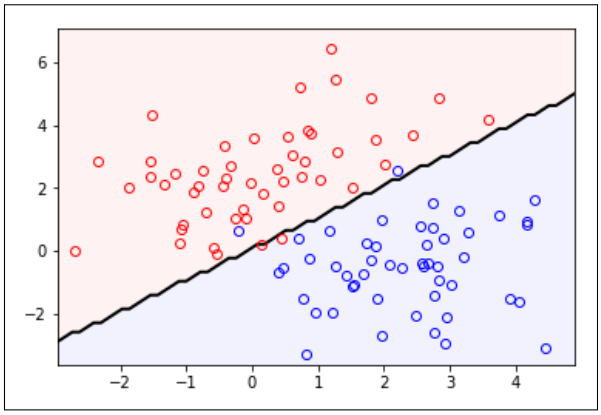


Figure 6: Prediction of Optimal Bayes Classifier

Implementation of Bayes Classifier

```
from scipy.stats import multivariate_normal
rv_xGivenBlue = multivariate_normal([2.5,0],
                                    [[2,1],[1,2]])
rv_xGivenRed = multivariate_normal([0,2.5],
                                   [[2,1],[1,2]])
Bayes_pred = []
for pos in np.c_[x_mesh.ravel(),y_mesh.ravel()]:
    p_xGivenBlue = rv_xGivenBlue.pdf(pos);
    p_xGivenRed = rv_xGivenRed.pdf(pos);
    if (p_xGivenBlue > p_xGivenRed):
        Bayes_pred.append(1)
    else:
        Bayes_pred.append(0)
Bayes_pred = np.array(Bayes_pred)
Bayes_pred = Bayes_pred.reshape(x_mesh.shape)
```

(c).

LDA Classifier

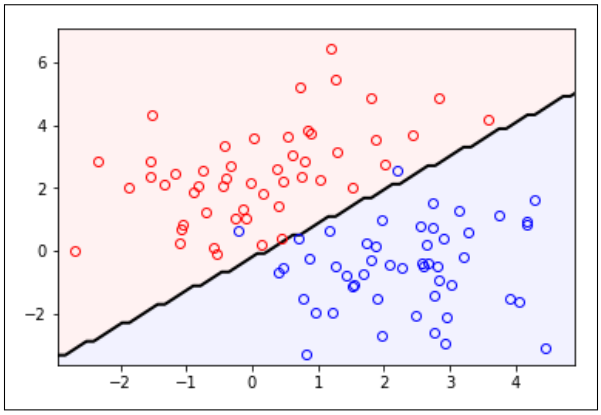


Figure 7: Prediction of LDA Classifier

LDA vs Bayes Classifier

The LDA classifier assumes a normal distribution with same covariance for classification. Since the inherent distribution in the dataspace is also Gaussian, the decision boundary between LDA and Bayes classifier appears similar. Also note that since the data points are classified with same covariance, a linear decision boundary is drawn.

(d).

QDA Classifier

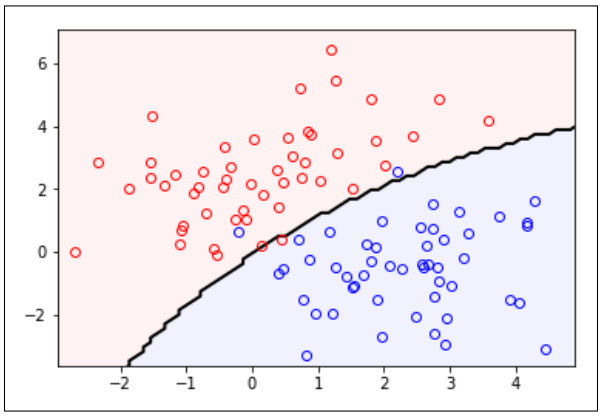


Figure 8: Prediction of QDA Classifier

QDA vs LDA Classifier

Since QDA assumes a different covariance for each class, the decision space in the QDA classifier is not linear and this is where it differs from LDA. LDA assumes similar covariance across classes.

(e).

Logistic Classifier

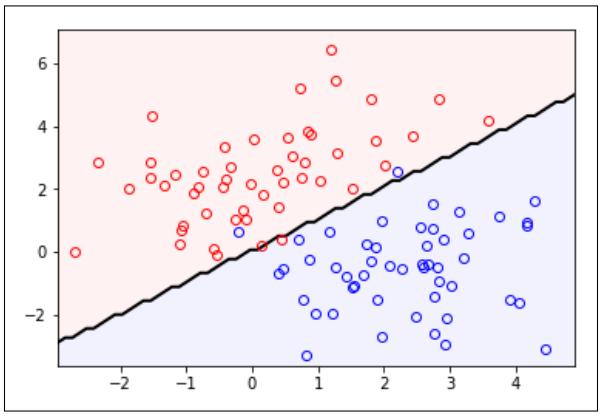


Figure 9: Prediction of Logistic Classifier

Logisitc vs LDA Classifier

The Logistic classifier, unlike LDA, is a regression algorithm. In that end, it produces an estimate of the probability immediately. It is more useful in classification of data with unknown probability. In this case, the regression algorithm learnt the underlying distribution to create a linear decision space.