Machine Learning

Homework 2+3

Karan Sunil Kumbhar
Id. - 12140860
BTech CSE
2025

CS550 Machine Learning

September 16, 2023

Problem 1
Ch4_Q4

Solution. (a) Answer:

Given

To calculate the fraction of available observations for prediction, we need to determine the fraction of the interval $[0, 1]$ that falls within the 10% range of X closest to the test observation.

Let's calculate:

a) To find the fraction of the available observations in this range, we calculate the length of the interval $[x-0.05, x+0.05]$ and divide it by the length of the entire interval $[0, 1]$.

So, on average, the fraction of available observations we will use to make the prediction can be represented as:

$$\text{Fraction of available observations} = \frac{\text{Length of } [x - 0.05, x + 0.05]}{\text{Length of } [0, 1]}$$

Now just calculate for

$$x = 0.6$$
$$\text{range of observations} = [0.55, 0.65]$$

Average,fraction of the available observations are, as we are using 10% range and also X is uniformly distributed

$$= \frac{\text{length}[0.55, 0.65]}{\text{length}[0,1]} \quad = \frac{0.1}{1} \times 100 \quad = \mathbf{10\%}$$

Problem 2
Ch4_Q4

Solution.

(b) Answer:

Given

To find the fraction of available observations used to make the prediction when we have two features X_1 and X_2 uniformly distributed on $[0, 1] \times [0, 1]$, and we use observations within 10% of the range of X_1 and X_2 closest to the test observation, we need to calculate the proportion of the total area that falls within these ranges.

Here's how we can calculate it:

a) Calculate the range of X_1 and X_2 within 10% of their respective total ranges:

   (a) For X_1, the range within 10% is $[0.45, 0.55]$.

   (b) For X_2, the range within 10% is $[0.45, 0.55]$.

b) Calculate the area of the square formed by these ranges:
   Area $= (0.55 - 0.45) \times (0.55 - 0.45) = 0.1 \times 0.1 = 0.01$

So, on average, we will use observations within 10% of the range of X_1 and within 10% of the range of X_2 closest to the test observation, which corresponds to approximately **1%** of the total area of the square formed by the ranges $[0, 1] \times [0, 1]$.

Problem 3
Ch4_Q4

Solution.

(c) Answer:

Given

To find the fraction of available observations used to make the prediction when we have $p = 100$ features, and we use observations within 10% of each feature's range closest to the test observation, we need to calculate the proportion of the total volume that falls within these ranges.

Here's how we can calculate it:

a) Calculate the range of each feature within 10% of its total range: For each feature, the range within 10% is $[0.45, 0.55]$ (10% of the total range).

b) Calculate the volume of the hypercube formed by these ranges:

$$\begin{aligned} \text{Volume} &= (0.55 - 0.45)^{100} \quad &= (0.1)^{100} \\ \text{Fraction} &= \frac{\text{Volume}}{\text{Total Volume}} \\ &= \frac{0.1^{100}}{1^{100}} \quad &= \{0.1^{100}\} \end{aligned}$$

This volume represents the fraction of available observations used to make the prediction. In practice, as p increases, the fraction of available observations used for prediction becomes exponentially small. The fraction of observations used for prediction decreases significantly as the number of features increases.

Problem 4
Ch4_Q4

Solution.

(d) Answer

a) **Part (a) - Single Feature ($p = 1$):** In this scenario, where we have only one feature, the curse of dimensionality is not a significant concern. We can effectively use observations that are within 10% of the feature's range from the test observation, resulting in a relatively large fraction of available observations for prediction. This is because the range along a single dimension is still manageable.

b) **Part (b) - Two Features ($p = 2$):** When we have two features, we find that we need to consider two separate 10% ranges for each feature. The fraction of available observations for prediction starts to decrease compared to the single-feature case. However, it's still possible to find observations within these ranges. The curse of dimensionality is not a severe issue in this case.

c) **Part (c) - High-Dimensional Space ($p = 100$):** In this scenario, with 100 features, we observe a significant problem. The fraction of available observations for prediction becomes extremely small. Due to the curse of dimensionality, the volume within 10% of each feature's range becomes tiny in high-dimensional space. This results in very few training observations "near" any given test observation.

Conclusion: As we increase the number of features, the fraction of available observations within a "neighborhood" around a test observation diminishes rapidly. In high-dimensional spaces, KNN struggles because there are very few training observations that are close or similar to any test observation. This makes it challenging to find meaningful neighbors for prediction. The curse of dimensionality severely limits the effectiveness of KNN in such scenarios.

Problem 5
Ch4_Q4

Solution.

(e) Answer

In general, for a hypercube centered around a test observation that contains, on average, 10% of the training observations, the length of each side of the hypercube can be calculated as follows:

length of each side of the p dimension hypercube is as follows-
consider follwing variables :-

V = total Volume , v = fractional volume

X = side of big hypercube, x = side of fraction hypercube

Calculations :-

$$\begin{aligned} X &= 1 \\ V &= X^p \\ &= 1^p \quad = 1 \\ v &= (10\%) \times V \\ &= (0.1) \times 1 \\ &= 0.1 \\ \text{also } v &= x^p \\ \therefore x &= v^{\frac{1}{p}} \quad = (0.1)^{\frac{1}{p}} \end{aligned}$$

a) For $p = 1$ (1-dimensional hypercube): The length of each side of the hypercube is simply 10% of the range along the single dimension. From part (a), we found that for $p = 1$, this is 0.1.

b) For $p = 2$ (2-dimensional hypercube):

$$x = (0.1)^{\frac{1}{2}} = 0.316$$

c) For $p = 100$ (100-dimensional hypercube): The fraction of available observations for prediction became very small, in this case

$$x = (0.1)^{\frac{1}{100}} = 0.977$$

Comment: The length of each side of the hypercube increases as the dimensionality (p) increases. In the case of the 100-dimensional hypercube, the side length is almost equal to

1, meaning that the hypercube covers almost the entire original space. This is opposite of KNN which demonstrates the "curse of dimensionality," where as the number of dimensions increases, we consider less observations.

Problem 6
Ch4_Q5

Solution.

a) **If the Bayes decision boundary is linear,** we expect **QDA** to perform better on the training set and **LDA** on the test set compared to **QDA**.

- **Training Set:** LDA assumes that the classes have the same covariance matrix (homoscedasticity) and estimates a single covariance matrix. QDA, on the other hand, estimates a separate covariance matrix for each class. As QDA is more flexible it will give more accuracy than LDA on train data
- **Test Set:** LDA is expected to generalize better to the test set as well. QDA, with its more flexible approach of estimating separate covariances, might overfit the training data when the true decision boundary is linear, leading to poorer performance on unseen test data.

b) **If the Bayes decision boundary is non-linear,** we expect **QDA** to perform better on the training and test set compared to **LDA**.

- **Training Set:** When the Bayes decision boundary is non-linear, LDA's assumption of a linear decision boundary is incorrect. LDA models the data with the assumption of equal covariance matrices for all classes, which is too restrictive for non-linear decision boundaries. QDA, on the other hand, can capture non-linear decision boundaries by estimating separate covariance matrices for each class.
- **Test Set:** On the test set,as here the non-linearity is present, So QDA is likely to outperform LDA because it can adapt to the non-linear patterns.

c) **In general, as the sample size n increases,** we expect **the test prediction accuracy of QDA will improve relative to LDA.**

- **Reasoning:** As the sample size n increases, both LDA and QDA models will have more data to learn from. However, QDA, being a more flexible model that allows for different covariance matrices for each class, can better capture complex relationships within the data, including non-linearities and variations in class-specific variances.
- **LDA's Limitation:** LDA, on the other hand, assumes equal covariance matrices for all classes, which can be restrictive when the data exhibits varying class-specific variances or non-linear decision boundaries. As the sample size increases, the limitations of LDA become more significant.

d) **False**.

As here the Bayes decision boundary is linear, Linear Discriminant Analysis (LDA) is expected to perform better on the test error rate compared to Quadratic Discriminant Analysis (QDA). This is due to the fact that LDA assumes that the class covariances are equal, meaning that the covariance matrix for each class is the same. This assumption simplifies the model and forces the decision boundary to be linear.

On the other hand, QDA allows for different covariance matrices for each class, making it more flexible and capable of modeling non-linear decision boundaries. However, when the true decision boundary is linear, the additional flexibility of QDA may lead to overfitting, especially if the sample size is small. This can result in a higher test error rate for QDA compared to LDA.

Problem 7

Chapter 4
Q 12

Solution.
Given

Given Model

$$\widehat{Pr}(Y = orange|X = x) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 x}}{1 + e^{\hat{\beta}_0+\hat{\beta}_1 x}}$$

Friend's Model

$$Pr(Y = \text{orange}|X = x) = \frac{e^{\hat{\alpha}_{\text{orange}0}+\hat{\alpha}_{\text{orange}1} x}}{e^{\hat{\alpha}_{\text{orange}0}+\hat{\alpha}_{\text{orange}1} x} + e^{\hat{\alpha}_{\text{apple}0}+\hat{\alpha}_{\text{apple}1} x}}$$

a) log-odds of orange versus apple in given model:

$$\begin{aligned}
\text{log odds(orange vs. apple)} &= \log\left(\frac{Pr(Y = \text{orange}|X = x)}{Pr(Y = \text{apple}|X = x)}\right) \\
&= \log\left(\frac{Pr(Y = \text{orange}|X = x)}{1 - Pr(Y = \text{orange}|X = x)}\right) \\
&= \log\left(\frac{\frac{e^{\hat{B}_0+\hat{B}_1 x}}{1+e^{\hat{B}_0+\hat{B}_1 x}}}{1 - \frac{e^{\hat{B}_0+\hat{B}_1 x}}{1+e^{\hat{B}_0+\hat{B}_1 x}}}\right) \\
&= \log\left(\frac{\frac{e^{\hat{B}_0+\hat{B}_1 x}}{1+e^{\hat{B}_0+\hat{B}_1 x}}}{\frac{1}{1+e^{\hat{B}_0+\hat{B}_1 x}}}\right) \\
&= \log\left(e^{\hat{B}_0+\hat{B}_1 x}\right) \\
&= \hat{B}_0 + \hat{B}_1 x
\end{aligned}$$

b) Friend's model (softmax formulation) yields:

$$\begin{aligned}
\text{log-odds(orange vs. apple)} &= \log\left(\frac{Pr(Y=\text{orange}|X=x)}{Pr(Y=\text{apple}|X=x)}\right) \\
&= \log\left(\frac{Pr(Y=\text{orange}|X=x)}{1-Pr(Y=\text{orange}|X=x)}\right) \\
&= \log\left(\frac{\frac{e^{\hat{\alpha}_{\text{orange0}}+\hat{\alpha}_{\text{orange1}}x}}{e^{\hat{\alpha}_{\text{orange0}}+\hat{\alpha}_{\text{orange1}}x}+e^{\hat{\alpha}_{\text{apple0}}+\hat{\alpha}_{\text{apple1}}x}}}{1-\frac{e^{\hat{\alpha}_{\text{orange0}}+\hat{\alpha}_{\text{orange1}}x}}{e^{\hat{\alpha}_{\text{orange0}}+\hat{\alpha}_{\text{orange1}}x}+e^{\hat{\alpha}_{\text{apple0}}+\hat{\alpha}_{\text{apple1}}x}}}\right) \\
&= \log\left(\frac{\frac{e^{\hat{\alpha}_{\text{orange0}}+\hat{\alpha}_{\text{orange1}}x}}{e^{\hat{\alpha}_{\text{orange0}}+\hat{\alpha}_{\text{orange1}}x}+e^{\hat{\alpha}_{\text{apple0}}+\hat{\alpha}_{\text{apple1}}x}}}{\frac{e^{\hat{\alpha}_{\text{apple0}}+\hat{\alpha}_{\text{apple1}}x}}{e^{\hat{\alpha}_{\text{orange0}}+\hat{\alpha}_{\text{orange1}}x}+e^{\hat{\alpha}_{\text{apple0}}+\hat{\alpha}_{\text{apple1}}x}}}\right) \\
&= \log\left(\frac{e^{\hat{\alpha}_{\text{orange0}}+\hat{\alpha}_{\text{orange1}}x}}{e^{\hat{\alpha}_{\text{apple0}}+\hat{\alpha}_{\text{apple1}}x}}\right) \\
&= (\hat{\alpha}_{\text{orange0}}-\hat{\alpha}_{\text{apple0}})+(\hat{\alpha}_{\text{orange1}}-\hat{\alpha}_{\text{apple1}})x
\end{aligned}$$

So, the log-odds of orange versus apple in your model is $\hat{\beta}_0+\hat{\beta}_1 x$. and in friend's model is $(\hat{\alpha}_{\text{orange0}}-\hat{\alpha}_{\text{apple0}})+(\hat{\alpha}_{\text{orange1}}-\hat{\alpha}_{\text{apple1}})x$

c) The coefficient estimates in Friend's model can be inferred by comparing the two models. In given model, $\hat{\beta}_0=2$ and $\hat{\beta}_1=-1$. To find the equivalent coefficients in Friend's softmax model, we can match the log odds expressions:

From the previous answer, we know that in given model, the log odds of orange versus apple is:

$$\text{log-odds(orange vs. apple)} = \hat{\beta}_0+\hat{\beta}_1 x$$

In Friend's model, the log odds of orange versus apple is:

$$\text{log odds(orange vs. apple)} = (\hat{\alpha}_{\text{orange0}}-\hat{\alpha}_{\text{apple0}})+(\hat{\alpha}_{\text{orange1}}-\hat{\alpha}_{\text{apple1}})x$$

Comparing these two expressions, we can equate the coefficients:

$$\begin{aligned}
\hat{\beta}_0 &= \hat{\alpha}_{\text{orange0}}-\hat{\alpha}_{\text{apple0}} \\
\hat{\beta}_1 &= \hat{\alpha}_{\text{orange1}}-\hat{\alpha}_{\text{apple1}}
\end{aligned}$$

Now, with $\hat{\beta}_0=2$ and $\hat{\beta}_1=-1$ from given model, we can calculate the equivalent coefficients for Friend's model:

$$\begin{aligned}
\hat{\alpha}_{\text{orange0}} &= 2+\hat{\alpha}_{\text{apple0}} \\
\hat{\alpha}_{\text{orange1}} &= -1+\hat{\alpha}_{\text{apple1}}
\end{aligned}$$

These equations represent the coefficient estimates in Friend's model in terms of your model's coefficients β_0 and β_1.

d) In Friend's model, the coefficient estimates are given as:

$$\hat{\alpha}_{\text{orange0}} = 1.2$$
$$\hat{\alpha}_{\text{orange1}} = -2$$
$$\hat{\alpha}_{\text{apple0}} = 3$$
$$\hat{\alpha}_{\text{apple1}} = 0.6$$

To find the equivalent coefficients in given logistic regression model, we can use the relationships established earlier:

$$\hat{\beta}_0 = \hat{\alpha}_{\text{orange0}} - \hat{\alpha}_{\text{apple0}}$$
$$\hat{\beta}_1 = \hat{\alpha}_{\text{orange1}} - \hat{\alpha}_{\text{apple1}}$$

Now, we can calculate the coefficients in your model:

$$\hat{\beta}_0 = 1.2 - 3 = -1.8$$
$$\hat{\beta}_1 = -2 - 0.6 = -2.6$$

So, in given logistic regression model, the coefficient estimates are $\hat{\beta}_0 = -1.8$ and $\hat{\beta}_1 = -2.6$.

e) To determine the fraction of the time you expect the predicted class labels from your model to agree with those from your friend's model, you can consider the probabilities of predicting each class in both models.

In your model, you have:

$$\Pr(Y = \text{orange}|\mathbf{X}) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x)}$$

And in your friend's model, you have:

$$\Pr(Y = \text{orange}|\mathbf{X}) = \frac{\exp(\hat{\alpha}_{\text{orange0}} + \hat{\alpha}_{\text{orange1}} x)}{\exp(\hat{\alpha}_{\text{orange0}} + \hat{\alpha}_{\text{orange1}} x) + \exp(\hat{\alpha}_{\text{apple0}} + \hat{\alpha}_{\text{apple1}} x)}$$

To find the agreement between the predicted class labels, you need to compare the predicted probabilities for each observation in both models. Specifically, you're interested in when both models predict the same class.

Let p_1 be the predicted probability of the orange class from your model and p_2 be the predicted probability of the orange class from your friend's model.

The fraction of the time both models predict the same class can be calculated as follows:

$$\begin{aligned}\text{Agreement Fraction} =& \Pr(\text{Y = orange in your model}) \times \Pr(\text{Y = orange in your friend's model}) \\ &+ \Pr(\text{Y = apple in your model}) \times \Pr(\text{Y = apple in your friend's model})\end{aligned}$$

You can calculate this fraction using the probabilities from both models. It represents the proportion of times your model's predictions agree with your friend's model when applied to the same test data set.

To calculate the fraction of the time both models predict the same class, we need to find the values of x for which the predicted probabilities from both models are the same. Let's set up the equations for both models and find the values of x that satisfy them.

In your model:

$$\Pr(Y = \text{orange}) = \frac{\exp(2 - x)}{1 + \exp(2 - x)}$$

$$\Pr(Y = \text{apple}) = \frac{1}{1 + \exp(2 - x)}$$

In your friend's model:

$$\Pr(Y = \text{orange}) = \frac{\exp(1.2 - 2x)}{\exp(1.2 - 2x) + \exp(3 + 0.6x)}$$

$$\Pr(Y = \text{apple}) = \frac{\exp(3 + 0.6x)}{\exp(1.2 - 2x) + \exp(3 + 0.6x)}$$

Now, we want to find the values of x for which:

$$\Pr(Y = \text{orange}) = \Pr(Y = \text{apple})$$

Let's equate the probabilities from both models and solve for x:

$$\frac{\exp(2 - x)}{1 + \exp(2 - x)} = \frac{\exp(1.2 - 2x)}{\exp(1.2 - 2x) + \exp(3 + 0.6x)}$$

Solving this equation for x will give us the values at which both models predict the same class. The fraction of the time they agree will depend on how many solutions there are and the range of x values considered.

Problem 8

Chapter 9
Q 2

Solution.

a) Equation of cureve

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

```
import numpy as np
import matplotlib.pyplot as plt

# Define a range for X1 and X2
X1 = np.linspace(-3, 3, 400)
X2 = np.linspace(-1, 5, 400)

# Create a grid of X1 and X2 values
X1, X2 = np.meshgrid(X1, X2)

# Define the equation for the curve
curve = (1 + X1)**2 + (2 - X2)**2 - 4

# Create a contour plot for the curve
plt.figure(figsize=(6, 6))
contour = plt.contour(X1, X2, curve, levels=[0], colors='blue')
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Curve: (1 + X1)^2 + (2 - X2)^2 = 4')
plt.grid(True)

# Show the plot
plt.show()
```

b) Plotting Two regions

$$\text{Region1} = > (1 + X_1)^2 + (2 - X_2)^2 > 4,$$

$$\text{Region2} = > (1 + X_1)^2 + (2 - X_2)^2 \leq 4,$$

```
import numpy as np
import matplotlib.pyplot as plt

# Define a range for X1 and X2
X1 = np.linspace(-3, 3, 400)
X2 = np.linspace(-1, 5, 400)

# Create a grid of X1 and X2 values
X1, X2 = np.meshgrid(X1, X2)

# Define the equation for the curve
curve = (1 + X1)**2 + (2 - X2)**2 - 4

# Create a contour plot for the curve
plt.figure(figsize=(6, 6))
contour = plt.contour(X1, X2, curve, levels=[0], colors='blue')
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Curve: (1 + X1)^2 + (2 - X2)^2 = 4')
plt.grid(False)

# Define the regions based on the inequalities
region1 = (1 + X1)**2 + (2 - X2)**2 > 4

circle = plt.Circle((-1, 2), 2, fill=False, color='red',
    linestyle='dashed', linewidth=2)
plt.gca().add_patch(circle)

# Shade the regions
plt.imshow(region1, extent=(-3, 3, -1, 5), origin='lower', cmap='
    Reds', alpha=0.5, aspect='auto', vmin=0, vmax=1)
plt.text(1, 4.3, '(1 + X1)^2 + (2 - X2)^2 > 4', fontsize=12,
    color='black', ha='center', va='center')
plt.imshow(~region1, extent=(-3, 3, -1, 5), origin='lower', cmap=
    'Greens', alpha=0.5, aspect='auto', vmin=0, vmax=1)
plt.text(-1, 2.2, '(1 + X1)^2 + (2 - X2)^2 <= 4', fontsize=12,
    color='black', ha='center', va='center')

plt.show()
```

c) Classification of Points

Point	Class
(0, 0)	Blue
(-1, 1)	Red
(2, 2)	Blue
(3, 8)	Blue

```
import numpy as np
import matplotlib.pyplot as plt

# Define a range for X1 and X2
X1 = np.linspace(-3, 4, 400)
X2 = np.linspace(-1, 10, 400)  # Adjusted the upper limit for
    better visibility

# Create a grid of X1 and X2 values
X1, X2 = np.meshgrid(X1, X2)

# Define the equation for the curve
curve = (1 + X1)**2 + (2 - X2)**2 - 4

# Create a contour plot for the curve
plt.figure(figsize=(7, 13))
contour = plt.contour(X1, X2, curve, levels=[0], colors='blue')
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Curve: (1 + X1)^2 + (2 - X2)^2 = 4')
plt.grid(False)

# Define the regions based on the inequalities
region1 = (1 + X1)**2 + (2 - X2)**2 > 4

# Create circular boundaries with a margin around the curve
circle = plt.Circle((-1, 2), 2, fill=False, color='red',
    linestyle='dashed', linewidth=2)
plt.gca().add_patch(circle)

# Shade the regions and add labels
plt.imshow(region1, extent=(-3, 4, -1, 10), origin='lower', cmap=
    'Reds', alpha=0.5, aspect='auto', vmin=0, vmax=1)
plt.text(0.5, 6, 'Region 1', fontsize=12, color='black', ha='
    center', va='center')
plt.imshow(~region1, extent=(-3, 4, -1, 10), origin='lower', cmap
    ='Greens', alpha=0.5, aspect='auto', vmin=0, vmax=1)
plt.text(-1, 2.2, 'Region 2', fontsize=12, color='black', ha='
    center', va='center')
```

```
# Define the points for classification
points = [(0, 0), (-1, 1), (2, 2), (3, 8)]

# Classify and plot the points
for point in points:
    x1, x2 = point
    if (1 + x1)**2 + (2 - x2)**2 > 4:
        label = f'Blue Class[{x1},{x2}]'
        plt.scatter(x1, x2, color='blue', label=label)
    else:
        label = f'Red Class[{x1},{x2}]'
        plt.scatter(x1, x2, color='red', label=label)
    plt.annotate(f'({x1}, {x2})', (x1, x2), textcoords="offset
    points", xytext=(0, 10), ha='center')

# Show the plot
plt.xlim(-3, 4)  # Adjusted the x-axis limits
plt.ylim(-1, 10)  # Adjusted the y-axis limits
plt.legend()
plt.show()
```

d) **(d) Argument**

The decision boundary given in part (c) is defined as:

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

This decision boundary is nonlinear when considered directly in terms of X_1 and X_2. However, when we expand and simplify the equation, it becomes linear in terms of X_1, X_1^2, X_2, and X_2^2.

Let's perform the expansion:

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

$$1 + 2X_1 + X_1^2 + 4 - 4X_2 + X_2^2 > 4$$

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

Now, if we consider new variables for X_1, X_2,X_1^2, X_2^2 as X_1, X_2, X_3, and X_4 respectively, we get:

$$2X_1 - 4X_2 + X_3 + X_4 + 1 > 0$$

This is a linear equation in terms of X_1, X_2,X_3 and X_4. Thus, while the original decision boundary is nonlinear in X_1 and X_2, it becomes linear when expressed in terms of X_1, X_1^2, X_2, and X_2^2, or equivalently, X_1, X_2,X_3 and X_4.

Problem 9

Chapter 9
Q 3

Solution.

a) Observations with Class Labels

Obs.	X_1	X_2	Y
1	3	4	Red
2	2	2	Red
3	4	4	Red
4	1	4	Red
5	2	1	Blue
6	4	3	Blue
7	4	1	Blue

```
import matplotlib.pyplot as plt

# Data
X1 = [3, 2, 4, 1, 2, 4, 4]
X2 = [4, 2, 4, 4, 1, 3, 1]
labels = ['Red', 'Red', 'Red', 'Red', 'Blue', 'Blue', 'Blue']

# Create a scatter plot
plt.scatter(X1, X2, c=labels, cmap='viridis', s=10, marker='o')

# Label points with their coordinates
num = 0
for i, txt in enumerate(labels):
    num+=1
    if(X2[i]<3.5):
        plt.annotate(f'({X1[i]}, {X2[i]})', (X1[i], X2[i]),
    textcoords="offset points", xytext=(0, 10), ha='center')
        plt.annotate(f'P{num}', (X1[i], X2[i]), textcoords="
    offset points", xytext=(0, -11), ha='center')
    else:
        plt.annotate(f'({X1[i]}, {X2[i]})', (X1[i], X2[i]),
    textcoords="offset points", xytext=(0, -15), ha='center')
        plt.annotate(f'P{num}', (X1[i], X2[i]), textcoords="
    offset points", xytext=(5, 4), ha='center')

# Set axis labels and title
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Observations with Class Labels')

# Display the plot
plt.grid(False)  # Remove grid lines
plt.show()
```

b) from graph we can observe that point P3 and P2 are the points where support vector of class red and P5 and P6 are the points of class blue from where support vector can pass

So the support vector of Class Red can be
P2 = [2,2], P3 = [4,4]

$$y_1 = m_1 x + b$$
$$y_1 = \frac{4-2}{4-2}x + b$$
$$y_1 = x + b \quad (1)$$
$$\text{also} \quad b = y_1 - x$$

Point $P_2[2,2]$ satisfy line (1)

$$b = 2 - 2$$
$$= 0$$

So line(1) can be written as

$$y_1 = x$$

Now the support vector of Class Blue can be
P5 = [2,1], P6 = [4,3]

$$y_2 = m_2 x + b$$
$$y_2 = \frac{3-1}{4-2}x + b$$
$$y_2 = x + b \quad (2)$$
$$\text{also} \quad b = y_2 - x$$

Point $P_5[2,1]$ satisfy line (2)

$$b = 1 - 2$$
$$= -1$$

So line(1) can be written as

$$y_2 = x - 1$$

Also seperating plane can be calcuated from line equation (1) and (2).As line 1 and line 2 are parallel, equation of seperating plane is

$$Y = x + \frac{(0 + (-1))}{2}$$
$$Y = x - 0.5 \quad (3)$$

Sketching seperating plane

```
import matplotlib.pyplot as plt
import numpy as np

# Data
X1 = [3, 2, 4, 1, 2, 4, 4]
X2 = [4, 2, 4, 4, 1, 3, 1]
labels = ['Red', 'Red', 'Red', 'Red', 'Blue', 'Blue', 'Blue']

# Create a scatter plot
plt.scatter(X1, X2, c=labels, cmap='viridis', s=10, marker='o')

# Label points with their coordinates
num = 0
for i, txt in enumerate(labels):
    num+=1
    if(X2[i]<3.5):
        plt.annotate(f'({X1[i]}, {X2[i]})', (X1[i], X2[i]),
    textcoords="offset points", xytext=(0, 10), ha='center')
        plt.annotate(f'P{num}', (X1[i], X2[i]), textcoords="
    offset points", xytext=(0, -11), ha='center')
    else:
        plt.annotate(f'({X1[i]}, {X2[i]})', (X1[i], X2[i]),
    textcoords="offset points", xytext=(0, -15), ha='center')
        plt.annotate(f'P{num}', (X1[i], X2[i]), textcoords="
    offset points", xytext=(5, 4), ha='center')

# Define the slope and intercept
slope = 1
intercept = -0.5

# Generate x values
x_values = np.linspace(1, 5, 500)
y_values = slope * x_values + intercept

# Plot the line
plt.plot(x_values, y_values, linestyle='--', color='green', label
    ='Seperating Plane')

# Set axis labels and title
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Observations with Class Labels')
plt.legend()

plt.grid(False)
plt.show()
```

c) Classification rules

1. **Class Red**

$$(-0.5) + X_1 - X_2 < 0$$

2. **Class Blue**

$$(-0.5) + X_1 - X_2 > 0$$

d)

```
import matplotlib.pyplot as plt
import numpy as np

# Data
X1 = [3, 2, 4, 1, 2, 4, 4]
X2 = [4, 2, 4, 4, 1, 3, 1]
labels = ['Red', 'Red', 'Red', 'Red', 'Blue', 'Blue', 'Blue']

# Create a scatter plot
plt.scatter(X1, X2, c=labels, cmap='viridis', s=10, marker='o')

# Label points with their coordinates
num = 0
for i, txt in enumerate(labels):
    num+=1
    if(X2[i]<3.5):
        plt.annotate(f'({X1[i]}, {X2[i]})', (X1[i], X2[i]),
    textcoords="offset points", xytext=(0, 10), ha='center')
        plt.annotate(f'P{num}', (X1[i], X2[i]), textcoords="
    offset points", xytext=(0, -11), ha='center')
    else:
        plt.annotate(f'({X1[i]}, {X2[i]})', (X1[i], X2[i]),
    textcoords="offset points", xytext=(0, -15), ha='center')
        plt.annotate(f'P{num}', (X1[i], X2[i]), textcoords="
    offset points", xytext=(5, 4), ha='center')

# Define the slope and intercept
slope = 1
intercept = -0.5

# Generate x values
x_values = np.linspace(1, 5, 500)
y_values = slope * x_values + intercept

# Plot the line
plt.plot(x_values, y_values, linestyle='--', color='green', label
    ='Seperating Plane')

# Set axis labels and title
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Observations with Class Labels')
plt.legend()

plt.grid(False)
plt.show()
```