VIETNAM NATIONAL UNIVERSITY
HCMC UNIVERSITY OF TECHNOLOGY

FINAL PROJECT

PROBLEM 1 EXERCISE

CLASS: CC01 - GROUP: 10 – SEMESTER 231

GVHD: PhD. Phung Thanh Huy

TEAM LIST

STT	ID	FULL NAME	% Project score	Project score	Note
1	2052154	Trần Minh Long			
2	2052537	Nguyễn Đăng Khoa			
3	2052073	Ngô Minh Đức			

HO CHI MINH CITY, 2023

REPORT GROUP 10 WORK

STT	ID	Full name	Member's task	Signature
1	2052154	Trần Minh Long		
2	2052537	Nguyễn Đăng Khoa		
3	2052073	Ngô Minh Đức		

TABLE OF CONTENT

1. Problem 1

Problem 1. Linear Regression (0.5 điểm)

Consider a linear regression model:

$$y^{(i)} = w_0 + \sum_{j=1}^{n} w_j x_j^{(i)}$$

Show that the given model can be represented as:

$$y^{(i)} - \bar{y} = \sum_{j=1}^{n} w_j \left(x_j^{(i)} - \bar{x}_j\right)$$

where $\bar{y}$ and $\bar{x}_j$ are the mean values of the dependent variable y and *independent variable* x_j respectively.

(Thereby, a possible way to remove w_0 in the original model is centering.)

Consider a linear regression model:

$$y^{(i)} = w_0 + \sum_{j=1}^{n} w_j x_j^{(i)} \quad (1)$$

Where:

w_0 is the intercept (the value of y when x=0)

w_j is the slope (the regression coefficient)

Consider the situation when we have just one

The intercept can be calculated by this formula:

$$w_0 = \frac{1}{n}\sum_{j=1}^{n} y - \sum_{j=1}^{n} w_j \bar{x}_j \quad (2)$$

From (1) and (2), we have:

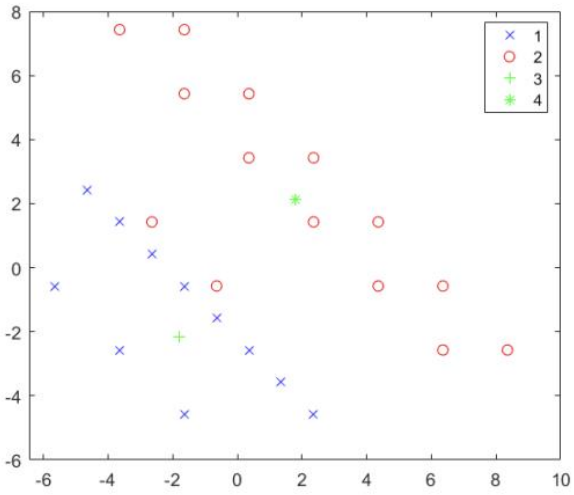
$$y^{(i)} = \frac{1}{n}\sum_{j=1}^{n} y - \sum_{j=1}^{n} w_j \bar{x}_j + \sum_{j=1}^{n} w_j x_j^{(i)}$$

$$\Leftrightarrow y^{(i)} - \bar{y} = \sum_{j=1}^{n} w_j \left(x_j^{(i)} - \bar{x}_j\right)$$

2. Problem 2

Problem 2. PCA. (2 điểm)

The given figure represents a dataset of 2 classes: Class 1 (o) and Class 2 (×). The marks (*) and (+) show the centers of the 2 classes.



The means values of the classes are:

$$\mu_1 = \begin{bmatrix} -1.8 \\ -2.1 \end{bmatrix}, \mu_2 = \begin{bmatrix} 1.8 \\ 2.1 \end{bmatrix}$$

The covariance matrix of the dataset:

$$\Sigma = \begin{bmatrix} 12.4 & -2.4 \\ -2.4 & 13.1 \end{bmatrix}$$

a) Calculate the mean value of the whole dataset regardless of the classes.

The mean value of the dataset with respect to the first feature:

$$\mu = \frac{\mu_1 \times 11 + \mu_2 \times 14}{25} = \frac{-1.8 \times 11 + 1.8 \times 14}{25} = 0.216$$

The mean value of the dataset with respect to the second feature:

$$\mu = \frac{\mu_1 \times 11 + \mu_2 \times 14}{25} = \frac{-2.1 \times 11 + 2.1 \times 14}{25} = 0.252$$

The mean value of the dataset:

$$\mu = \begin{bmatrix} 0.216 \\ 0.252 \end{bmatrix}$$

b) Calculate the eigen values and eigen vectors of covariance matrix

The covariance matrix of the dataset:

$$S = \begin{bmatrix} 12.4 & -2.4 \\ -2.4 & 13.1 \end{bmatrix}$$

The characteristic equation of the covariance matrix is:

$$\det(S - \lambda I) = 0$$

$$\begin{bmatrix} 12.4 - \lambda & -2.4 \\ -2.4 & 13.1 - \lambda \end{bmatrix} = 0 \Leftrightarrow (12.4 - \lambda)(13.1 - \lambda) - 2.4^2 = 0$$

$$\lambda^2 - 25.5\lambda + 156.68 = 0 \Leftrightarrow \lambda = 15.1754 \; or \; \lambda = 10.3246$$

Compute the eigen vectors:

$$\begin{bmatrix} 12.4 - \lambda & -2.4 \\ -2.4 & 13.1 - \lambda \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Leftrightarrow \begin{bmatrix} (12.4 - \lambda)v_1 & -2.4v_2 \\ -2.4v_1 & (13.1 - \lambda)v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Leftrightarrow \frac{v_1}{2.4} = \frac{v_2}{12.4 - \lambda} = t$$

$$\Leftrightarrow v_1 = 2.4t, v_2 = (12.4 - \lambda)t$$

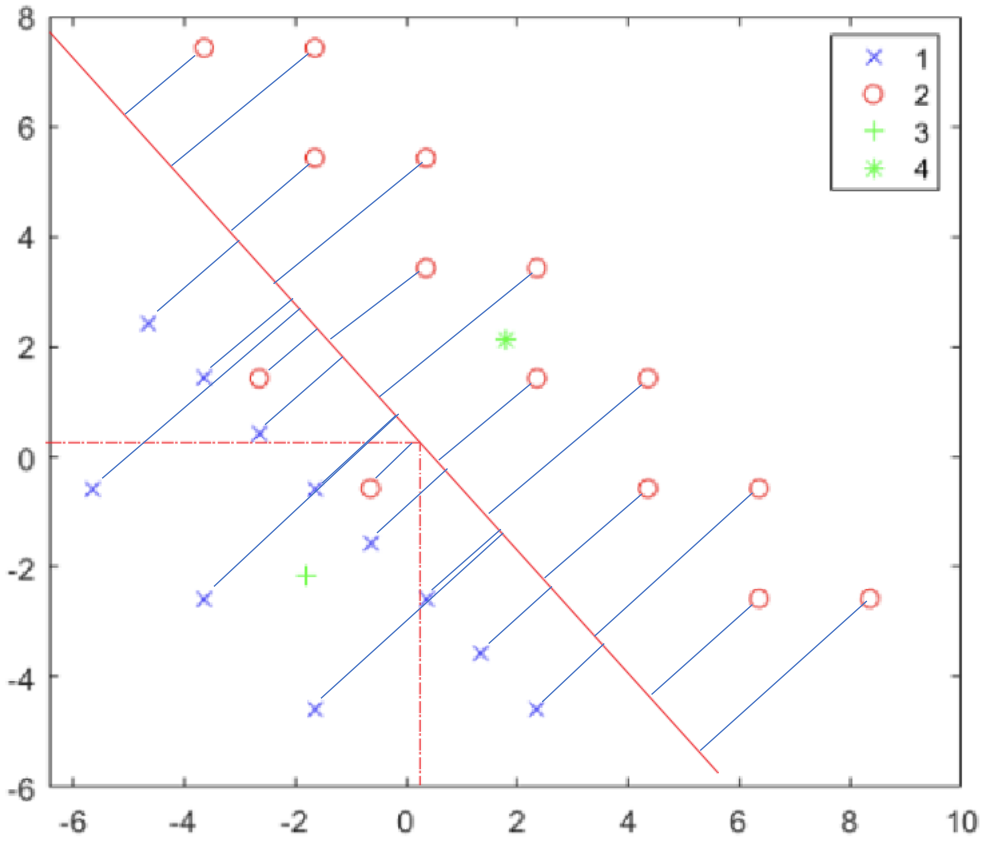
We choose $\lambda = 15.1754 \; and \; t = 1$

$$\Leftrightarrow V_1 = \begin{bmatrix} 2.4 \\ -2.7754 \end{bmatrix} = \begin{bmatrix} \frac{2.4}{\sqrt{2.4^2 + (-2.7754)^2}} \\ \frac{-2.7754}{\sqrt{2.4^2 + (-2.7754)^2}} \end{bmatrix} = \begin{bmatrix} 0.6541 \\ -0.7564 \end{bmatrix}$$

We choose $\lambda = 10.3246 \;\; and \; t = 1$

$$\Leftrightarrow V_2 = \begin{bmatrix} 2.4 \\ 2.0754 \end{bmatrix} = \begin{bmatrix} \frac{2.4}{\sqrt{2.4^2 + (2.0754)^2}} \\ \frac{2.0754}{\sqrt{2.4^2 + (2.0754)^2}} \end{bmatrix} = \begin{bmatrix} 0.7564 \\ 0.6541 \end{bmatrix}$$

c) Represent PC1 on the figure and project the data to PC1 (calculation is not necessary)


8
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6
8
10
1
2
3
4

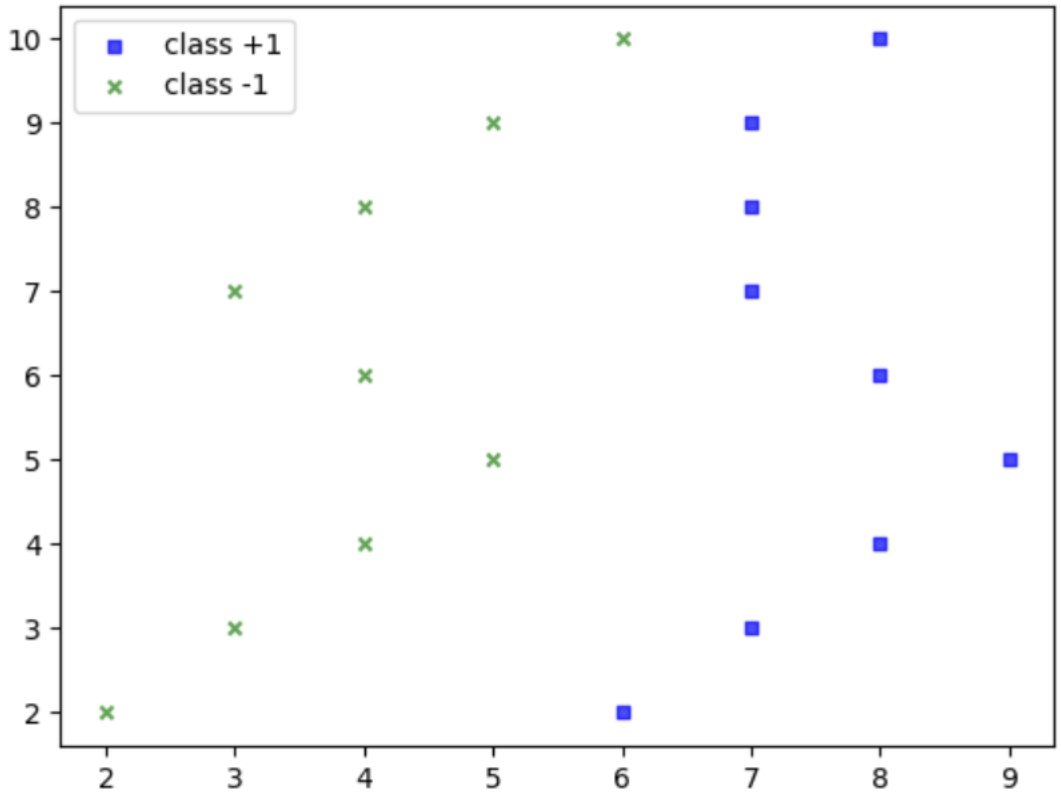
3. Problem 3

Problem 3. Gaussian naïve Bayes (1.5 điểm)

Provide a binary classification dataset as below:

$$\text{Class -1}: \begin{bmatrix} 2 & 2 \\ 3 & 3 \\ 4 & 4 \\ 5 & 5 \\ 4 & 6 \\ 3 & 7 \\ 4 & 8 \\ 5 & 9 \\ 6 & 10 \end{bmatrix} \qquad \text{Class +1}: \begin{bmatrix} 6 & 2 \\ 7 & 3 \\ 8 & 4 \\ 9 & 5 \\ 8 & 6 \\ 7 & 7 \\ 7 & 8 \\ 7 & 9 \\ 8 & 10 \end{bmatrix}$$

a) Represent the data on a 2D figure (manually)



b) Use Gaussian naïve Bayes to classify the dataset

From the dataset, we have:

$$P(Class+1) = \frac{9}{18} = 0.5$$

$$P(Class-1) = \frac{9}{18} = 0.5$$

In Class+1 class:

$$Mean(Feature\ 1) = \frac{6+7+8+9+8+7+7+7}{8} = 7.375$$

$$Variance(Feature\ 1) = \frac{(6-7.375)^2 + \cdots + (7-7.375)^2 + (7-7.375)^2}{8-1} = 0.839$$

$$Mean(Feature\ 2) = \frac{2+3+4+5+6+7+8+9}{8} = 5.5$$

$$Variance(Feature\ 2) = \frac{(2-5.5)^2 + (3-5.5)^2 + \cdots + (9-5.5)^2}{8-1} = 6$$

Following that, we can calculate the mean and variance of other classes.

Class	Mean(Feature 1)	Var(Feature 1)	Mean(Feature 2)	Var(Feature 2)
Class+1	7.375	0.839	5.5	6
Class-1	3.75	1.071	5.5	6

For the first test case (Feature 1=6, Feature 2=10):

$$Posterior(Class+1) = \frac{P(Class+1)P(Feature\ 1|Class+1)P(Feature\ 2|Class+1)}{Evidence} = \frac{0.5 \times 0.141 \times 0.03}{1} = 2.115 \times 10^{-3}$$

$$Posterior(Class-1) = \frac{P(Class-1)P(Feature\ 1|Class-1)P(Feature\ 2|Class-1)}{Evidence} = \frac{0.5 \times 0.036 \times 0.03}{1} = 5.4 \times 10^{-4}$$

Therefore, test 1 belongs to class +1 which is different from the actual result.

For the second test case (Feature 1=8, Feature 2=10):

$$Posterior(Class + 1)$$
$$= \frac{P(Class + 1)P(Feature\ 1|Class + 1)P(Feature\ 2|Class + 1)}{Evidence}$$
$$= \frac{0.5 \times 0.345 \times 0.03}{1} = 5.175 \times 10^{-4}$$

$$Posterior(Class - 1)$$
$$= \frac{P(Class - 1)P(Feature\ 1|Class - 1)P(Feature\ 2|Class - 1)}{Evidence}$$
$$= \frac{0.5 \times 8.39 \times 10^{-5} \times 0.03}{1} = 1.26 \times 10^{-6}$$

We can conclude that Test 2 is Class+1.

From each P, we can calculate using formula:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$$

μ: is the mean

σ^2: is the variance

c) (Advanced) If Bernoulli naïve Bayes is used to classify the dataset, what should be changed from (b)?

Bernoulli naïve Bayes is used when the features are binary, meaning they can only take on 2 values (0 or 1). BNB calculates the probability of each feature being 1 or 0 to each class and then uses these probabilities to estimate the class posterior probability.

Therefore, we can follow these steps for classifications using BNB:

- Discretize continuous features using techniques like binning or thresholding.
- Treat the discretized features as binary inputs for BNB.
- Estimate the probability of each feature being 1 for each class.
- Use these probabilities to calculate the class posterior probability for each data point.

4. Problem 4

Problem 4. naïve Bayes (1 điểm)

Giving a dataset to determine whether the fruit is a banana or not:

ID	Color	Form	Origin	Banana
1	yellow	oblong	imported	yes
2	yellow	round	domestic	no
3	yellow	oblong	imported	no
4	brown	oblong	imported	yes
5	brown	round	domestic	no
6	green	round	imported	yes
7	green	oblong	domestic	no
8	red	round	imported	no

The vector independent variable: $x = [x_1, x_2, x_3]^T$ where x_1: Color, x_2: Form, x_3: Origin. Label: yes (is banana) or no (is not banana)

In naïve Bayes, the Posteori ($P(x|y)$):

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

1. Use the provided dataset to calculate: Priori ($P(y)$), $P(x)$, and Posteori $P(x|y)$
2. With a new observation $x = [yellow, round, imported]$, predict if it is a banana or not.

a) Priori of Label

$$P(yes) = \frac{3}{8} = 0.375$$

$$P(no) = \frac{5}{8} = 0.625$$

Posteori of Color

	Yes	No
yellow	$\frac{1}{3}$	$\frac{2}{5}$
brown	$\frac{1}{3}$	$\frac{1}{5}$
green	$\frac{1}{3}$	$\frac{1}{5}$
red	0	$\frac{1}{5}$

Posteori of Form

	Yes	No
oblong	$\frac{2}{3}$	$\frac{2}{5}$
round	$\frac{1}{3}$	$\frac{3}{5}$

Posteori of Origin

	Yes	No
imported	1	$\frac{2}{5}$
domestic	0	$\frac{3}{5}$

b) With $x = [\boldsymbol{yellow}, \boldsymbol{round}, \boldsymbol{imported}]$, the probability that x is banana is

$P(yes|yellow, round, imported)$

$= P(yes)P(yellow \mid yes)P(round \mid yes)P(imported \mid yes)$

$$= 0.375 \times \frac{1}{3} \times \frac{1}{3} \times 1$$

$$= \frac{1}{24} = 0.0417$$

The probability that x is not banana is

$P(no|yellow, round, imported)$

$= P(no)P(yellow \mid no)P(round \mid no)P(imported \mid no) = 0.625 \times \frac{2}{5} \times \frac{3}{5} \times \frac{2}{5}$

$= \frac{3}{50} = 0.06$

$\Rightarrow x$ is predicted to be not banana

5. Problem 5

a) Loss function

Decision boundary:

$$\mathrm{g(x)} = w_0 + w^T x$$

With the decision boundary, have the distance function from closest data points to hyperplane:

$$\text{margin} = \frac{\mathrm{y} \times |w_0 + w^T x|}{\|\mathrm{w}\|_2}$$

Where:

$\|\mathrm{w}\|_2 = \sqrt{\sum_{i=1}^{d} w^2{}_i}$, d *is dimension of space*

Then, the optimization problem in SVM is to find w_0, w with maximum margin (prevent noise or new data points).

$$(\mathrm{w}, w_0) = \underset{w, w_0}{arg\ max} \left\{ \underset{n}{min}\ \frac{\mathrm{y} \times |w_0 + w^T x|}{\|\mathrm{w}\|_2} \right\} = \underset{w, w_0}{arg\ max} \left\{ \frac{1}{\|\mathrm{w}\|_2} \underset{n}{min} \mathrm{y} \times |w_0 + w^T x| \right\}$$

With every n:

$$\mathrm{y} \times |w_0 + w^T x| \geqslant 1$$

To find w_0, w with the constraints as below:

$$(\mathrm{w}, w_0) = \underset{w, w_0}{arg\ max} \left\{ \frac{1}{\|\mathrm{w}\|_2} \right\}$$

Subject to: $\mathrm{y} \times |w_0 + w^T x| \geqslant 1$, with n = 1, 2, 3,…, N

Finally, since maximizing $\frac{1}{\|\mathrm{w}\|_2}$ is the same as minimizing $\|\mathrm{w}\|_2{}^2 / 2$, can re-express the optimization problem as:

$$(\mathrm{w}, w_0) = \underset{w, w_0}{arg\ min} \left\{ \frac{1}{2} \|\mathrm{w}\|_2{}^2 \right\}$$

Subject to: $\mathrm{y} \times |w_0 + w^T x| \geqslant 1$, with n = 1, 2, 3,…, N

For non-seperatable cases, need to add ξ_i (slack variable) dictates the degree to which the constraint on the ith datapoint can be violated (certain training points will be allowed to be within the margin).

The new optimization problem can be expressed as:

$$\underset{w, w_0, \xi_{1:N}}{min} \left\{ \sum \xi_i + \frac{1}{2} \|\mathrm{w}\|_2{}^2 \right\}$$

$$\text{Subject to: } \mathrm{y} \times |w_0 + w^T x| \geqslant 1 - \xi_i, \xi_i \geqslant 0$$

$$\text{If } \mathrm{y} \times |w_0 + w^T x| \geqslant 1, \text{then } \xi_i = 0$$

$$\text{If } \mathrm{y} \times |w_0 + w^T x| < 1, \text{then } \xi_i = 1 - \mathrm{y} \times |w_0 + w^T x|$$

After that, we got:

$$\xi_i = max(0, 1 - \mathrm{y} \times |w_0 + w^T x|)$$

From the above equation, the Hinge loss is defined as $l_{hinge}(y, \hat{y}) = max(0, 1 - y\hat{y})$:

$$\underset{w, w_0}{min} \left\{ \sum_j l_{hinge}\,(y, w_0 + w^T x) \right\}$$

b) The slack variable ξ_i

Data points are correctly classified have $\xi_i = 0$, while data points inside the margin, but correctly classified, have $0 < \xi_i < 1$. $Data\ points\ misclassified\ have\ \xi_i \geqslant 1$.

For example: true label = 1, prediction = -1 (misclassified)

$$\xi_i = l_{hinge}(1, -1) = max(0, 1 - 1(-1)) = 2 \geqslant 1$$

true label = 1, prediction = 1 (correctly classified)

$$\xi_i = l_{hinge}(1, 1) = max(0, 1 - 1(1)) = 0$$

c) SVM for multiclassification

Support Vector Machines (SVMs) were originally designed for binary classification tasks, where the goal is to separate two classes. However, SVMs can also be extended to handle multiclass classification problems through various approaches. Here are two common methods:

One-vs-All (One-vs-Rest): In this approach, a separate SVM is trained for each class, treating it as the positive class while considering all other classes as the negative class. For example, if there are N classes, N separate SVMs are trained. During prediction, each SVM outputs a confidence score or distance from the decision boundary for each data point. The class with the highest score is then assigned as the predicted class.

One-vs-One: In this approach, a separate SVM is trained for each pair of classes. For N classes, this results in N * (N-1) / 2 SVMs. During prediction, each SVM votes for a class, and the class with the most votes is assigned as the predicted class. This approach is also known as the pairwise classification strategy.

d) SVM in Scikit-learn library

LinearSVC:

penalty: {'l1', 'l2'}, default='l2'

Specifies the norm used in the penalization. The 'l2' penalty is the standard used in SVC. The 'l1' leads to coef_ vectors that are sparse.

loss: {'hinge', 'squared_hinge'}, default='squared_hinge'

Specifies the loss function. 'hinge' is the standard SVM loss (used e.g. by the SVC class) while 'squared_hinge' is the square of the hinge loss. The combination of penalty='l1' and loss='hinge' is not supported.

dual: "auto" or bool, default=True

Select the algorithm to either solve the dual or primal optimization problem. Prefer dual=False when n_samples > n_features. dual= "auto" will choose the value of the

parameter automatically, based on the values of n_samples, n_features, loss, multi_class and penalty. If n_samples < n_features and optimizer supports chosen loss, multi_class and penalty, then dual will be set to True, otherwise it will be set to False.

tol: float, default=1e-4

Tolerance for stopping criteria.

C: float, default=1.0

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive.

multi_class{'ovr', 'crammer_singer'}, default='ovr'

Determines the multi-class strategy if y contains more than two classes. "ovr" trains n_classes one-vs-rest classifiers, while "crammer_singer" optimizes a joint objective over all classes. While crammer_singer is interesting from a theoretical perspective as it is consistent, it is seldom used in practice as it rarely leads to better accuracy and is more expensive to compute. If "crammer_singer" is chosen, the options loss, penalty and dual will be ignored.

fit_intercept: bool, default=True

Whether or not to fit an intercept. If set to True, the feature vector is extended to include an intercept term: [x_1, ..., x_n, 1], where 1 corresponds to the intercept. If set to False, no intercept will be used in calculations (i.e. data is expected to be already centered).

intercept_scaling: float, default=1.0

When fit_intercept is True, the instance vector x becomes [x_1, ..., x_n, intercept_scaling], i.e. a "synthetic" feature with a constant value equal to intercept_scaling is appended to the instance vector. The intercept becomes intercept_scaling * synthetic feature weight. Note that liblinear internally penalizes the intercept, treating it like any other term in the feature vector. To reduce the impact of the regularization on the intercept, the intercept_scaling parameter can be set to a value

greater than 1; the higher the value of intercept_scaling, the lower the impact of regularization on it. Then, the weights become [w_x_1, ..., w_x_n, w_intercept*intercept_scaling], where w_x_1, ..., w_x_n represent the feature weights and the intercept weight is scaled by intercept_scaling. This scaling allows the intercept term to have a different regularization behavior compared to the other features.

class_weight: dict or 'balanced', default=None

Set the parameter C of class i to class_weight[i]*C for SVC. If not given, all classes are supposed to have weight one. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y)).

verbose: int, default=0

Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in liblinear that, if enabled, may not work properly in a multithreaded context.

random_state: int, RandomState instance or None, default=None

Controls the pseudo random number generation for shuffling the data for the dual coordinate descent (if dual=True). When dual=False the underlying implementation of LinearSVC is not random and random_state has no effect on the results. Pass an int for reproducible output across multiple function calls. See Glossary.

max_iter: int, default=1000

The maximum number of iterations to be run.

SVC:

C: float, default=1.0

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

kernel{'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'} or callable, default='rbf'

Specifies the kernel type to be used in the algorithm. If none is given, ‘rbf’ will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples). For an intuitive visualization of different kernel types see Plot classification boundaries with different SVM Kernels.

degree: int, default=3

Degree of the polynomial kernel function (‘poly’). Must be non-negative. Ignored by all other kernels.

gamma{‘scale’, ‘auto’} or float, default=’scale’

Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’:

if gamma='scale' (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,

if ‘auto’, uses 1 / n_features

if float, must be non-negative.

coef0: float, default=0.0

Independent term in kernel function. It is only significant in ‘poly’ and ‘sigmoid’.

shrinking: bool, default=True

Whether to use the shrinking heuristic. See the User Guide.

probability: bool, default=False

Whether to enable probability estimates. This must be enabled prior to calling fit, will slow down that method as it internally uses 5-fold cross-validation,
and predict_proba may be inconsistent with predict. Read more in the User Guide.

tol: float, default=1e-3

Tolerance for stopping criterion.

cache_size: float, default=200

Specify the size of the kernel cache (in MB).

class_weight: dict or 'balanced', default=None

Set the parameter C of class i to class_weight[i]*C for SVC. If not given, all classes are supposed to have weight one. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data as n_samples / (n_classes * np.bincount(y)).

verbose: bool, default=False

Enable verbose output. Note that this setting takes advantage of a per-process runtime setting in libsvm that, if enabled, may not work properly in a multithreaded context.

max_iter: int, default=-1

Hard limit on iterations within solver, or -1 for no limit.

decision_function_shape{'ovo', 'ovr'}, default='ovr'

Whether to return a one-vs-rest ('ovr') decision function of shape (n_samples, n_classes) as all other classifiers, or the original one-vs-one ('ovo') decision function of libsvm which has shape (n_samples, n_classes * (n_classes - 1) / 2). However, note that internally, one-vs-one ('ovo') is always used as a multi-class strategy to train models; an ovr matrix is only constructed from the ovo matrix. The parameter is ignored for binary classification.

break_ties: bool, default=False

If true, decision_function_shape='ovr', and number of classes > 2, predict will break ties according to the confidence values of decision_function; otherwise the first class among the tied classes is returned. Please note that breaking ties comes at a relatively high computational cost compared to a simple predict.

random_state: int, RandomState instance or None, default=None

Controls the pseudo random number generation for shuffling the data for probability estimates. Ignored when probability is False. Pass an int for reproducible output across multiple function calls. See Glossary.

When I set SVC(kernel = “linear”) is equivalent to LinearSVC.

6. Problem 6

a) Confusion matrix

Confusion matrix is a table that summarizes the performance of a classification model by displaying the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions. It is a useful tool for evaluating the performance of a classification model and understanding the types of errors it makes.

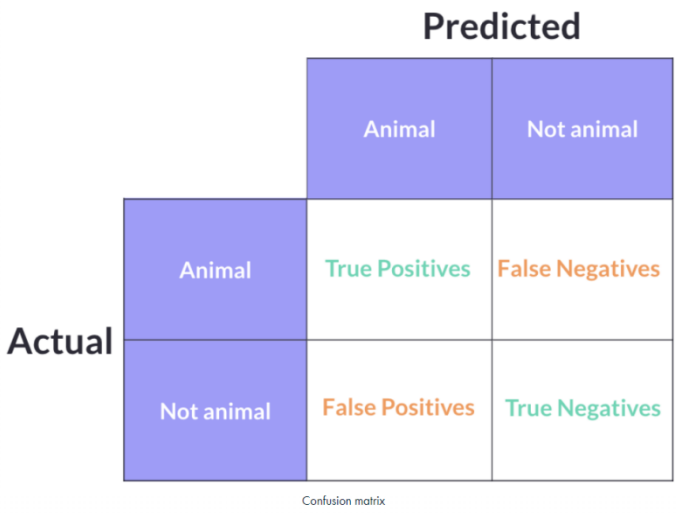


Figure 1: Confusion matrix

For example: has the six images to classify image is animal or not animal

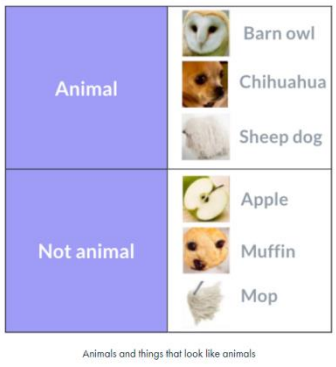
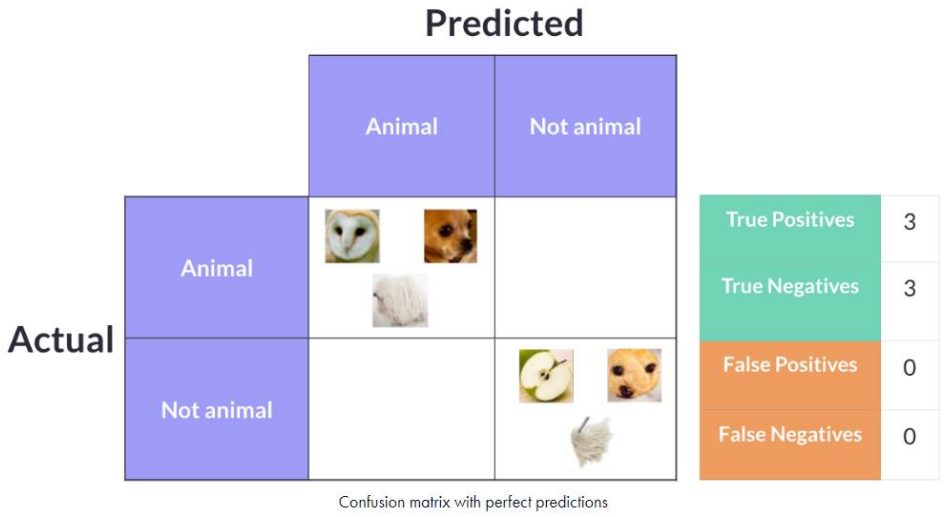


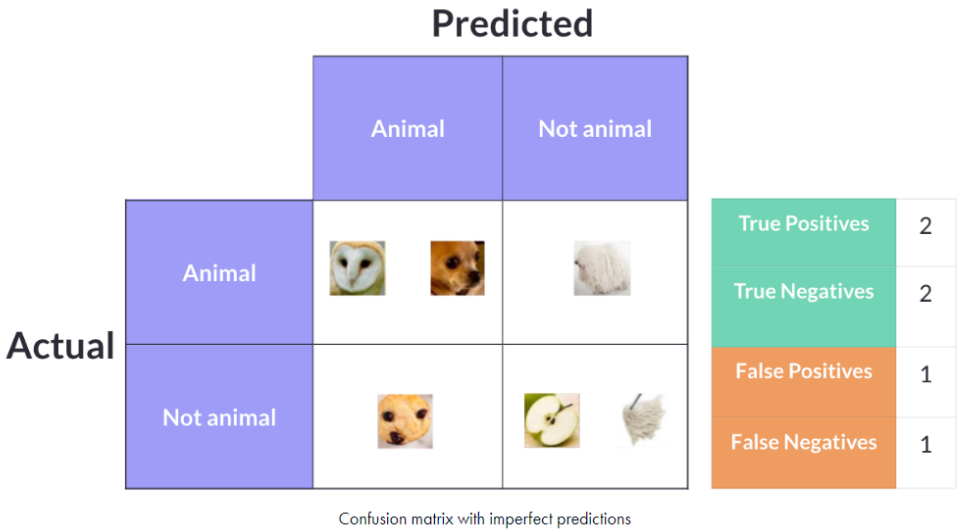
Figure 2: Animal and not animal images

The below confusion matrix shows 3 true positives and true negatives, 0 false positives and false negatives



Confusion matrix with perfect predictions

The below confusion matrix shows 2 true positives and true negatives, 1 false positives and false negatives



Confusion matrix with imperfect predictions

b) Accuracy, precision, recall, F1-score, precision and recall trade-off

Accuracy: accuracy measures the overall correctness of the model's predictions. It is the ratio of correctly predicted instances to the total number of instances in the dataset. Accuracy is calculated using the formula:

$$accuracy = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ negatives + false\ positives}$$

Figure 3: Accuracy formula

Precision: precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive. It focuses on the accuracy of positive predictions. Precision is calculated using the formula:

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

Figure 4: Precision formula

Recall: recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive instances out of all actual positive instances. It focuses on the ability of the model to identify positive instances. Recall is calculated using the formula:

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Figure 5: Recall formula

F1 score: the F1 score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance by considering both precision and recall. The F1 score is calculated using the formula:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 6: F1-score formula

For example: the below confusion matrix shows 3 true positives and false positives, 0 true negatives and false negatives. Can calculate accuracy - 50%, precision - 50%, recall - 100% and F1-score - 67%.

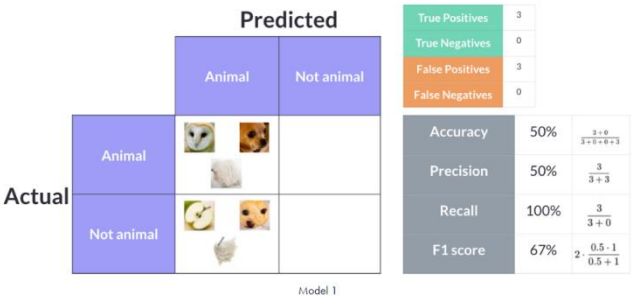


Figure 7: Accuracy, precision, recall and F1-score calculation

Precision and recall trade-off: refers to the relationship between precision and recall in a classification model. It highlights the fact that increasing one metric often leads to a decrease in the other. In many classification tasks, there is a trade-off between precision and recall because the model's decision threshold determines the balance between these two metrics. The decision threshold is a value that determines the classification boundary, where instances above the threshold are predicted as positive, and those below are predicted as negative.

The choice between precision and recall depends on the specific problem and the relative importance of false positives and false negatives in the application.

For example: in a spam email classification task, high precision is desired to minimize false positives (legitimate emails being classified as spam), while in a disease diagnosis task, high recall is crucial to minimize false negatives (missing actual cases of the disease).

c) ROC curve

The Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classification model. It illustrates the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at various classification thresholds:

True Positive Rate (TPR): Also known as sensitivity, recall, or hit rate, the TPR is the ratio of correctly predicted positive instances to the total number of actual positive instances. It represents the model's ability to correctly identify positive instances.

False Positive Rate (FPR): The FPR is the ratio of incorrectly predicted negative instances to the total number of actual negative instances. It represents the model's tendency to incorrectly classify negative instances as positive.

The ROC curve is created by plotting the TPR against the FPR at different classification thresholds. The classification threshold determines the point at which the model decides whether a prediction should be classified as positive or negative. By varying this threshold, we can calculate the TPR and FPR at each point and plot them on the graph.

The ROC curve typically ranges from (0,0) to (1,1). The diagonal line from (0,0) to (1,1) represents the performance of a random classifier, where the TPR is equal to the FPR. A point above the diagonal line indicates better-than-random performance, while a point below the line indicates worse-than-random performance.

The ideal position for an ROC curve is in the top-left corner, representing a high TPR and a low FPR. A model with an ROC curve that closely hugs the top-left corner indicates high discriminatory power, where it can effectively separate positive and negative instances.

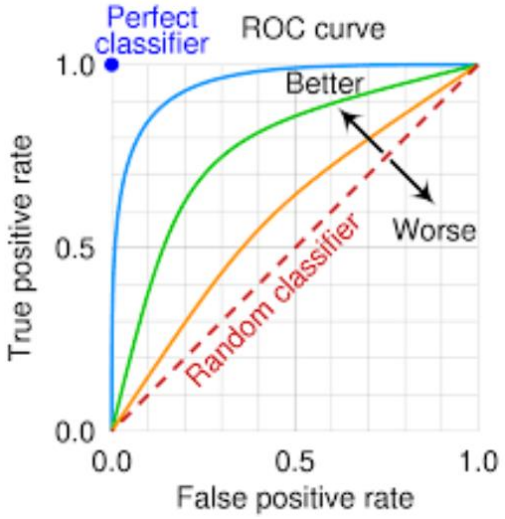


Figure 8: ROC curve

d) AUC (Area Under the ROC Curve)

The Area Under the ROC Curve (AUC) is a quantitative measure of the overall performance of the classifier. It represents the probability that a randomly chosen

positive instance will be ranked higher than a randomly chosen negative instance by the classifier. The AUC ranges from 0 to 1, with a higher value indicating better performance. An AUC of 0.5 suggests a random classifier, while an AUC of 1 indicates a perfect classifier.

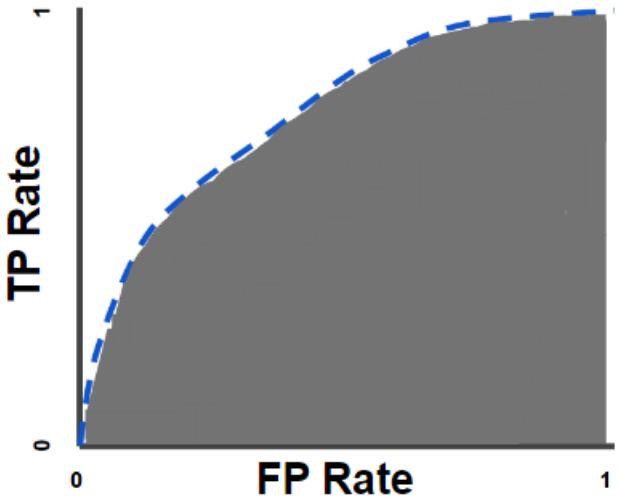


Figure 9: AUC

7. Problem 7

a) Confusion matrix

Because the threshold is 0.5, if the prediction probability is higher than 0.5, the sample is belong to class 1 and vice versa. Have the below table:

ID	True class	Prediction	Prediction
1	0	0.33	0
2	0	0.27	0
3	0	0.11	0
4	1	0.38	0
5	1	0.17	0
6	1	0.63	1
7	1	0.62	1
8	1	0.33	0
9	0	0.15	0
10	0	0.57	1

Figure 10: Prediction table

		Predict	
		1	**0**
Actual	**1**	2 (True Positives)	3 (False Negatives)
	0	1 (False Positives)	4 (True Negatives)

b) Accuracy, precision, recall and F1 score calculation

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{2 + 4}{2 + 4 + 1 + 3} = 0.6 = 60\%$$

$$\textbf{Precision} = \frac{TP}{TP + TN} = \frac{2}{2 + 4} = 0.33 = 33.33\%$$

$$\textbf{Recall} = \frac{TP}{TP + FN} = \frac{2}{2 + 3} = 0.4 = 40\%$$

$$\textbf{F1 score} = 2 \times \frac{precision \times recall}{precision + recall} = 2 \times \frac{0.33 \times 0.4}{0.33 + 0.4} = 0.36 = 36\%$$

c) Draw a ROC curve

Threshold	TPR	FPR
0.11	1	1
0.17	1	0.6
0.27	0.8	0.6
0.33	0.8	0.4
0.38	0.6	0.2
0.57	0.4	0.2
0.62	0.4	0
0.63	0.2	0
> 0.63	0	0

After that, use calculated TPR, FPR to draw ROC curve.

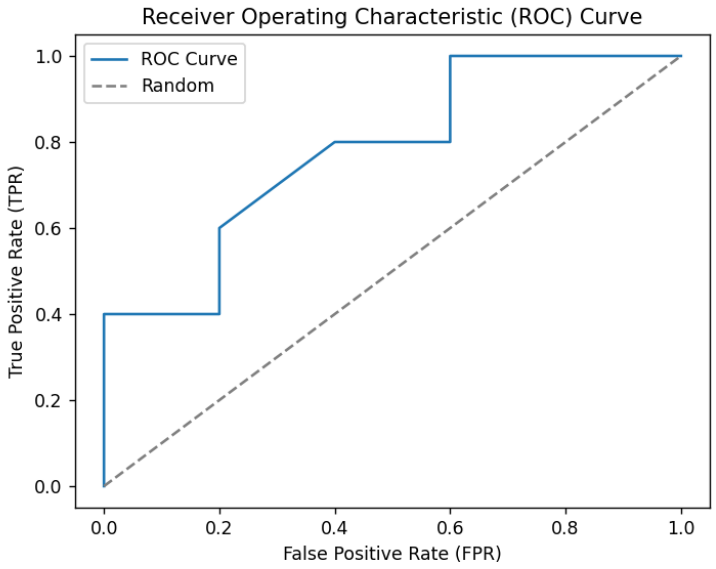


Figure 11: ROC curve

d) AUC

To calculate AUC (Area under the ROC curve), use Trapezoidal Rule, TPR, FPR and thresholds.

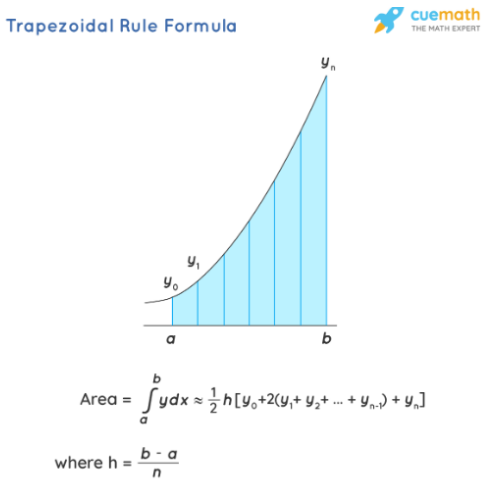


Figure 12: Trapezoidal Rule Formula

I got the AUC of 0.72

REFERENCES

[1] Purva Huilgol. 2023. *Precision and Recall | Essential Metrics for Machine Leaning.*

https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/

[2] Google Developers. 2022. *Classification: ROC Curve and AUC.*

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[3] Scikit-learn. 2007. *sklearn.svm.LinearSVC.*

https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[4] Scikit-learn. 2007. *sklearn.svm.SVC.*

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[5] Machine learning Cơ bản. 2017. *Bài 19: Support Vector Machine.*

https://machinelearningcoban.com/2017/04/09/smv/

[6] LAVANYA S. 2022. *Gaussian Naive Bayes Algorithm for Credit Risk Modelling.*

https://www.analyticsvidhya.com/blog/2022/03/gaussian-naive-bayes-algorithm-for-credit-risk-modelling/

[7] Machine learning Cơ bản. 2017. *Bài 27: Principal Component Analysis (phần 1/2).*

https://machinelearningcoban.com/2017/06/15/pca/

[8] Machine learning Cơ bản. 2017. *Bài 28: Principal Component Analysis (phần 2/2).*

https://machinelearningcoban.com/2017/06/21/pca2/

[9] Machine learning Cơ bản. 2016. *Bài 3: Linear Regression.*

https://machinelearningcoban.com/2016/12/28/linearregression/