Data Mining & Machine Learning – Assignment 2
(20% of total grade)

* If two homework submissions are found to be similar to each other, both submissions will receive 0.
* Homework solutions must be submitted through Canvas. If you have multiple files, please include all files as one zip file.
* For coding assignments, it is recommended to use **Jupytor notebook** and submit your **.ipynb** file.
* Answers with **math expressions** and **graphs** can be handwritten and scanned.
* If you find any **typo/error** in the assignment, let me know.

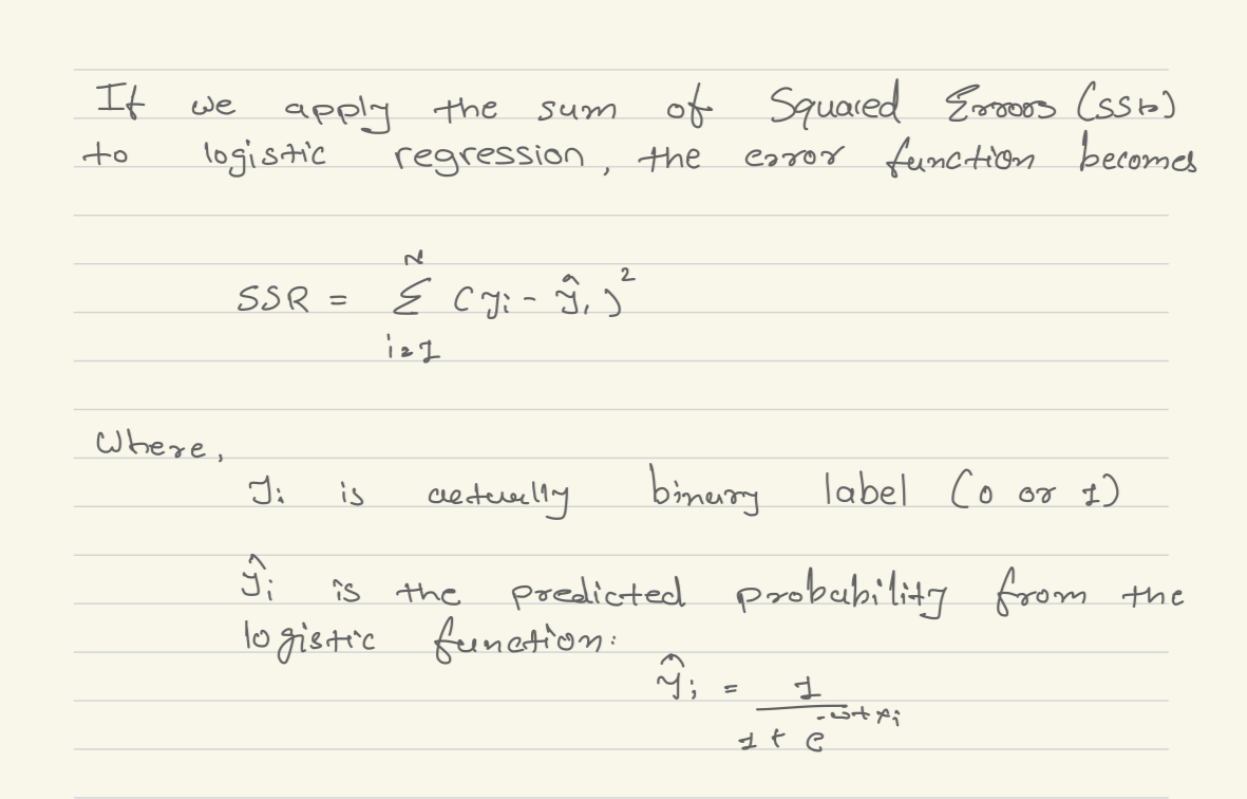
1. [3 pts/ea] Logistic regression uses cross-entropy as the error function.
1) Linear regression uses Sum Squared Error(SSR). Explain why Logistic regression uses cross-entropy.

Linear regression uses the Sum of Squared Error (SSR) because it predicts continuous values. The model minimizes the difference between the expected and actual values by squaring the errors, which works well for a constant output.

On the other hand, logistic regression predicts probabilities for binary classification (0 or 1). The output is not continuous but a probability, and the goal is to minimize the difference between the actual class labels (either 0 or 1) and the predicted probabilities. Using Cross-Entropy Loss (also known as Log Loss) makes more sense for probabilistic outputs because it penalizes incorrect predictions more effectively by taking the logarithm of the probability.

The cross-entropy loss is more aligned with the probabilistic nature of logistic regression, ensuring a proper fit and a convex loss function that facilitates finding a global minimum.

2) Show the error function of logistic regression using Sum of Squared Error(SSR)



In logistic regression, the predicted value is a probability between 0 and 1, but using SSR for probabilities creates challenges:

The SSR error function is **non-convex** for logistic regression, meaning it can have multiple local minima, making it hard to find the optimal solution.

The SSR function does not penalize the misclassification of probabilities as effectively as cross-entropy does.

Thus, SSR is not ideal for logistic regression due to its probabilistic output.

2. [3 pts] Explain using the reason linear regression as a classifier is very sensitive to outliers.

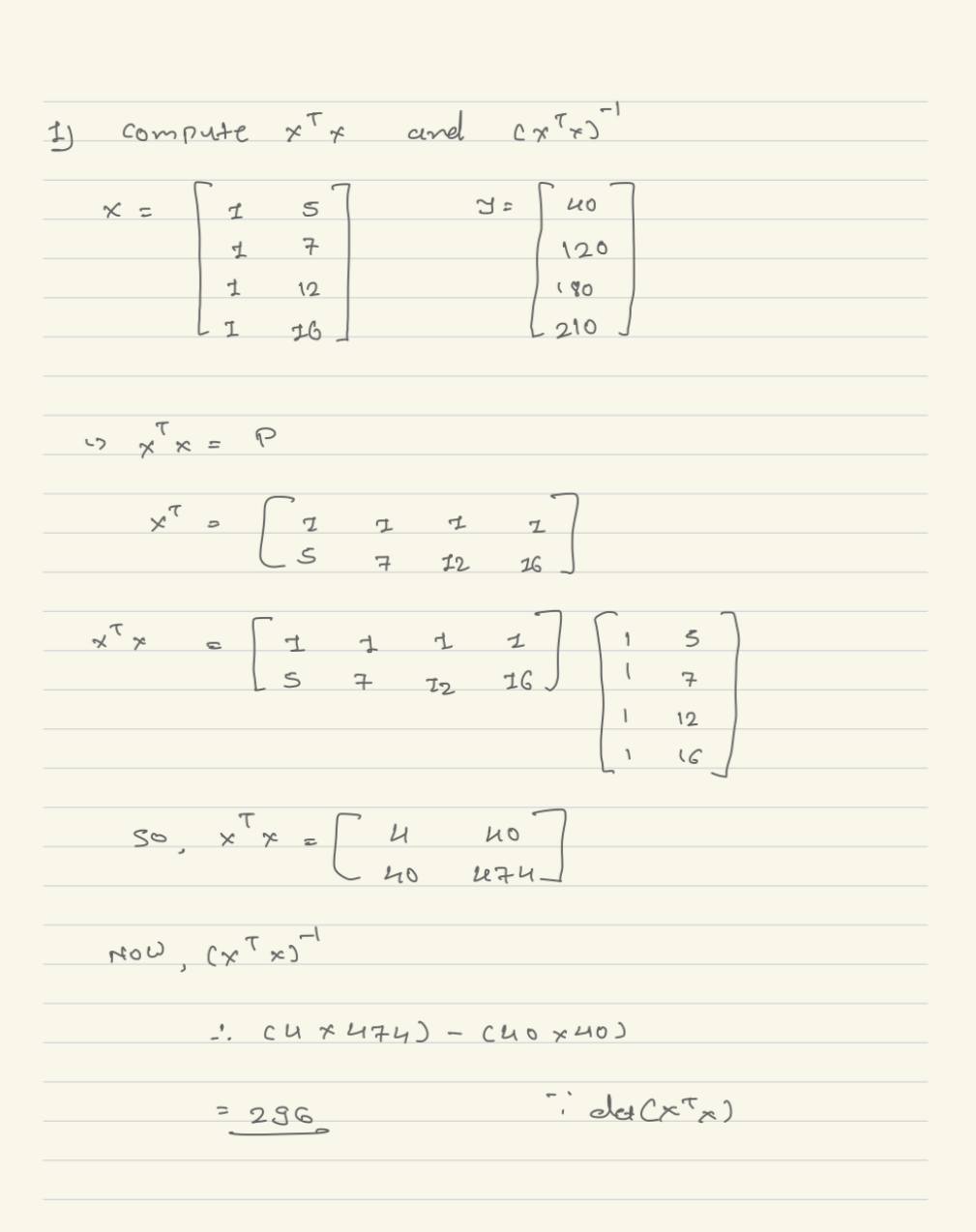
Linear regression models are sensitive to outliers because they minimize the sum of squared errors (SSE), which gives more weight to larger errors due to the squaring of residuals. When an outlier is present, the difference between the predicted and actual value becomes large, and squaring it amplifies this effect, causing the regression line to shift significantly to accommodate the outlier.

In classification tasks, linear regression is particularly unsuitable because outliers can distort the decision boundary, leading to poor classification performance. Since linear regression assumes continuous

outcomes, it doesn't handle categorical labels well, and outliers in the data can heavily influence the slope and intercept of the decision boundary, misclassifying the majority of the data points.

This is why logistic regression, which constrains the output to probabilities between 0 and 1, is more resilient against outliers in classification tasks.

3. [3 pts/ea] (Refer to p. 19-20 in linear regression slides.) Using the data
(x,y)=[(5, 40), (7, 120) (12, 180) (16, 210)]
1) compute $\mathrm{X}^{\mathrm{T}}X$ and $\left(\mathrm{X}^{\mathrm{T}}X\right)^{-1}$



$$(x^T x)^{-1}$$

$$= \frac{1}{296} \begin{bmatrix} 474 & -40 \\ -40 & 4 \end{bmatrix}$$

$$(x^T x)^{-1} = \begin{bmatrix} 1.6014 & -0.1351 \\ -0.1351 & 0.0135 \end{bmatrix}$$

2) compute $\hat{w}_0$ and $\hat{w}_1$.

2)

$$\hat{w} = (x^T x)^{-1} x^T y$$

$$x^T y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 5 & 7 & 12 & 16 \end{bmatrix} \begin{bmatrix} 40 \\ 120 \\ 180 \\ 210 \end{bmatrix}$$

$$x^T y = \begin{bmatrix} 550 \\ 5500 \end{bmatrix}$$

$$\hat{w} = \begin{bmatrix} 1.6014 & -0.1351 \\ -0.1351 & 0.0135 \end{bmatrix} \begin{bmatrix} 550 \\ 5500 \end{bmatrix}$$

$$\hat{w}_0 = (1.6014 \times 550) + (-0.1351 \times 5500)$$

$\hat{w}_0 = 880.77 - 743.05$

$\hat{w}_0 = 137.72$

$\hookrightarrow \hat{w}_1 = (-0.1351 \times 550) + (0.0135 \times 5500)$

$\hat{w}_1 = -74.31 + 74.25$

$\hat{w}_1 = -0.06$

4. [5 pts] Logistic regression is to maximize the following formula.

$$\prod_{i=1}^{N} p_1(x_i; w)^{y_i} p_0(x_i; w)^{1-y_i}$$

If we change above formula to below, does it still work? Explain exactly what is going to happen.

$$\prod_{i=1}^{N} p(x_i; w)^{y_i}$$

The original formula represents the **likelihood function** that logistic regression maximizes during training. It captures the probability of correctly classifying all data points given their respective labels. For each data point i, the contribution to the likelihood is:

•**When** y_i = 1**:**
The likelihood term becomes:

p_1(x_i; w)^1 \cdot p_0(x_i; w)^0 = p_1(x_i; w)

This is the probability that the data point belongs to **class 1**.

•**When** y_i = 0**:**
The likelihood term becomes:

p_1(x_i; w)^0 \cdot p_0(x_i; w)^1 = p_0(x_i; w) = 1 - p_1(x_i; w)

This represents the probability that the data point belongs to **class 0**.

Thus, the original formula models the **joint likelihood** by multiplying the probabilities for all data points. This ensures that the model considers both classes (1 and 0) during training and attempts to fit the data accordingly.

Analysis of the Modified Formula:

In the modified formula, the term p_0(x_i; w)^{1 - y_i} is **removed**. This leads to the following:

•**When** y_i = 1**:**
The modified formula is identical to the original:

p(x_i; w)^1 = p_1(x_i; w)

So, for data points with y_i = 1, the modified formula behaves correctly.

•**When** y_i = 0**:**
The modified formula becomes:

p(x_i; w)^0 = 1

This means that, for any data point with y_i = 0, the model **ignores the probability** of the point belonging to class 0. As a result, the likelihood of predicting class 0 correctly is no longer considered.

Consequence of the Change:

1.**Ignoring Class 0:**
With the removal of the class 0 term, the model pays attention only to data points with y_i = 1. As a result, it **ignores all data points where** y_i = 0 during training.
2.**Biased Model:**
The modified formula can lead to a model that becomes highly accurate in predicting class 1 but performs poorly at predicting class 0. This **imbalance** makes the model **biased** and less effective, especially when the dataset contains both classes.

In summary, by omitting the term for class 0, the model loses the ability to properly consider the likelihood of correctly classifying data points in class 0, leading to **poor performance** and **bias** towards class 1.

5. [3 pts] Logistic regression with many discrete values uses one-vs-all approach. Explain why.

When logistic regression is applied to a **multi-class classification problem** (where the target variable can take multiple discrete values, e.g., \(y \in \{0, 1, 2, \dots, K\}\)), the **one-vs-all (OvA)** or **one-vs-rest** approach is commonly used. Here's why this approach is necessary and effective:

Why Use One-vs-All for Multi-Class Logistic Regression?

Logistic Regression is Binary by Nature:

•Logistic regression is designed to solve **binary classification** tasks (e.g., classifying data into two categories like 0 vs. 1).

•For multi-class problems, where the target variable has more than two possible values, a strategy is required to extend logistic regression to handle these additional classes.

How the One-vs-All Approach Works:

•In the **OvA approach**, we train **K binary classifiers** (one for each class).

•Each classifier considers the current class as the **positive class** (e.g., "Class 1 vs. All Others").

•All other classes are treated as the **negative class** in that specific classifier.

•During prediction, each classifier outputs a probability score. The final predicted class is the one with the **highest probability** across all the classifiers.

Advantages of One-vs-All:

•**Simplicity:** It is easy to implement since it involves training multiple binary classifiers.

•**Efficiency:** It works well when the number of classes is small to moderate.

•**Interpretability:** Each binary classifier focuses on separating one class from the rest, making the model's behavior easier to understand.

Other Multi-Class Approaches:

•**One-vs-One Approach:** Trains a binary classifier for each pair of classes, which can become computationally expensive when there are many classes.

•**Softmax (Multinomial Logistic Regression):** Softmax generalizes logistic regression for multi-class problems but can be more complex to train and interpret compared to OvA.

6. Refer to Page No 8-12 in feature selection slides. A feature X has three values X1, X2 and X3. Using ANOVA F-test we want to test whether we will accept the feature X or reject it.

X1	X2	X3
8	5	7
8	6	6
10	6	7
7	4	7
10	8	9
$\overline{x_1}$	$\overline{x_2}$	$\overline{x_3}$
s_1^2	s_2^2	s_3^2

Critical Value = 3.89

1) [6 pts] compute F-value

In the given dataset there are three group

$x_1 = [8, 8, 10, 7, 10]$
$x_2 = [5, 6, 6, 4, 8]$
$x_3 = [7, 6, 7, 7, 9]$

Now, calculate the mean

$$\bar{x}_1 = \frac{8+8+10+7+10}{5} = 8.6$$

like this $\bar{x}_2 = 5.8$

$\bar{x}_3 = 7.2$

While calculating overall mean

$\bar{x} = 7.1$

Now calculating the SSB

$$SSB = n \sum_{i=1}^{3} (\bar{x}_i - \bar{x})^2$$

where $n = 5$

$$SSB = 5[(8.6 - 7.1)^2 + (5.8 - 7.1)^2 + (7.2 - 7.1)^2]$$
$$= 19.75$$

Now calculating SSW

following the eqn $= \sum_{i=1}^{3} \sum_{j=1}^{5} (x_{ij} - \bar{x}_i)^2$

for group $x_1 = 7.2$

$x_2 = 8.8$

$x_3 = 4.8$

$SSW = 7.2 + 8.8 + 4.8 = 20.8$

∴ Calculating mean square

$$MSB = \frac{SSB}{df_B} = \frac{19.75}{2} = 9.875$$

$$MSW = \frac{SSW}{df_B} = \frac{20.8}{12} = 1.733$$

∴ F value is

$$F = \frac{MSB}{MSW} = \frac{9.875}{1.733} \simeq \underline{5.7}$$

2) [3 pts] make decision using F-value and critical value.

- **Critical value**: 3.89 (given)
- **F-value**: 5.7

Since the calculated F-value 5.7 > 3.89, we **reject the null hypothesis**.

The F-value is greater than the critical value, so we reject the null hypothesis. This means that at least one of the group means is significantly different from the others, suggesting that the feature X is important and should be accepted.

7. [3 pts] Given D and d defined in p. 7 in PCA slides, explain why minimizing difference between D and d is equivalent to maximizing the variance of the projected data.

In **PCA (Principal Component Analysis)**, the objective is to reduce the number of dimensions while retaining the essential information about the **variance** in the data.

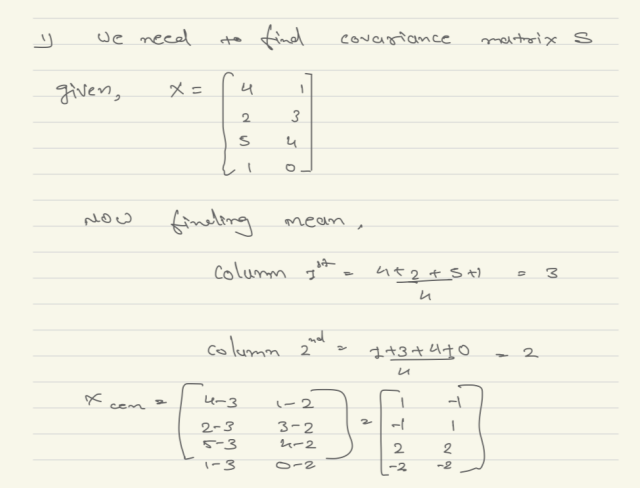
We compare the **distances between points** in the original high-dimensional space (D) with their distances after being projected into a lower-dimensional space (d). The closer these distances (d) are to the original (D), the better the projection preserves the structure of the original data.

By **minimizing the difference** between D and d, we **maximize the variance** in the lower-dimensional space, ensuring that the projection captures as much valuable information from the original data as possible.

8. [4 pts/ea] PCA

Given four sample points $X = \begin{bmatrix} 4 & 1 \\ 2 & 3 \\ 5 & 4 \\ 1 & 0 \end{bmatrix}$

1) compute covariance matrix S



Now,

$$S = \frac{1}{n-1} X_{cen}^{T} \cdot X_{cen}$$

$$= \frac{1}{3} \begin{bmatrix} 1 & -1 & 2 & -2 \\ -1 & 1 & 2 & -2 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \\ 2 & 2 \\ -2 & -2 \end{bmatrix}$$

$$= \frac{1}{3} \begin{bmatrix} 10 & 8 \\ 8 & 8 \end{bmatrix}$$

$$\therefore S = \begin{bmatrix} 10/3 & 8/3 \\ 8/3 & 8/3 \end{bmatrix}$$

2) compute eigenvalues λ_1 and λ_2

2)

λ_1 & λ_2 - we looking for ?

$$\therefore |S - \lambda I| = 0$$

$$\begin{bmatrix} 10/3 - \lambda & 8/3 \\ 8/3 & 8/3 - \lambda \end{bmatrix} = 0$$

$$\therefore \left(\frac{10}{3} - \lambda\right)\left(\frac{8}{3} - \lambda\right) - \left(\frac{8}{3}\right)^2$$

$$\therefore \frac{80}{9} - 6\lambda + \lambda^2 = \frac{64}{9} = \frac{16}{9} - 6\lambda + \lambda^2$$

$$\lambda^2 - 6\lambda + \frac{16}{9} = 0$$

$$9\lambda^2 - 54\lambda + 16 = 0$$

$$\therefore \lambda = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\therefore \lambda_1 = 5.687$$

$$\therefore \lambda_2 = 0.313$$

3) compute eigenvectors

3)

Now, Take $\lambda_1 = 5.687$,

$$\begin{bmatrix} 10/3 - 5.687 & 8/3 \\ 8/3 & 8/3 - 5.687 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

$$\begin{bmatrix} -2.020 & 2.667 \\ 2.667 & -3.354 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

$$v_2 = \frac{2.020}{2.667} \qquad v_1 = 0.757\, v_1$$

$$\therefore \quad v_1 = \begin{bmatrix} 0.752 \\ 0.662 \end{bmatrix}$$

Now, Take $\lambda_2 = 0.313$

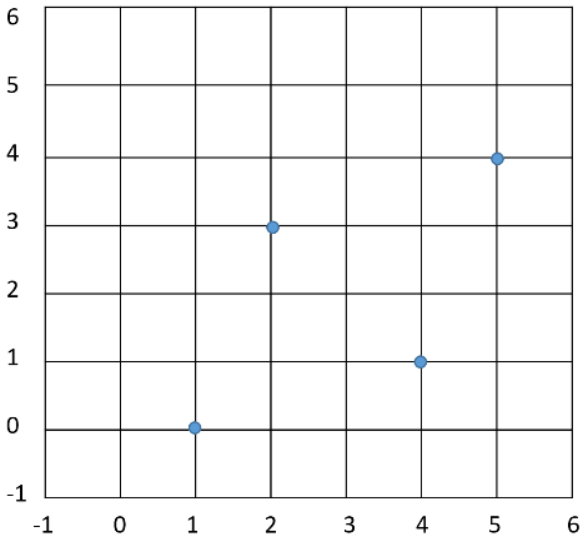
$$\begin{bmatrix} 10/3 - 0.313 & 8/3 \\ 8/3 & 8/3 - 0.313 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

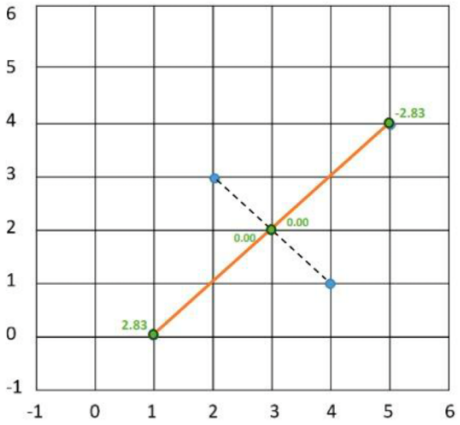
$$\begin{bmatrix} 2.987 & 2.667 \\ 2.667 & 2.354 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = 0$$

After Normalization

$$\therefore v_2 = \begin{bmatrix} -0.662 \\ 0.750 \end{bmatrix}$$

4) The plot below shows the sample points from X. Draw the principal component direction (as a line) and the projections of all four sample points onto the principal direction. Show the projected point with its principal coordinate value exactly.





9. [3 pts/ea] In Autoencoder, explain what happens in the following cases.
1) Autoencoder network with linear activation functions

When an autoencoder uses **linear activation functions** in all layers, it becomes mathematically **equivalent to Principal Component Analysis (PCA)**.

This is because both the encoder and decoder perform only **linear transformations** on the input data, limiting the model's ability to capture **linear relationships** between features.

As a result, the autoencoder projects the data into a lower-dimensional space just like PCA. However, it cannot model **nonlinear patterns**, restricting its effectiveness for more complex datasets.

2) The number of hidden nodes is greater than the number of input nodes

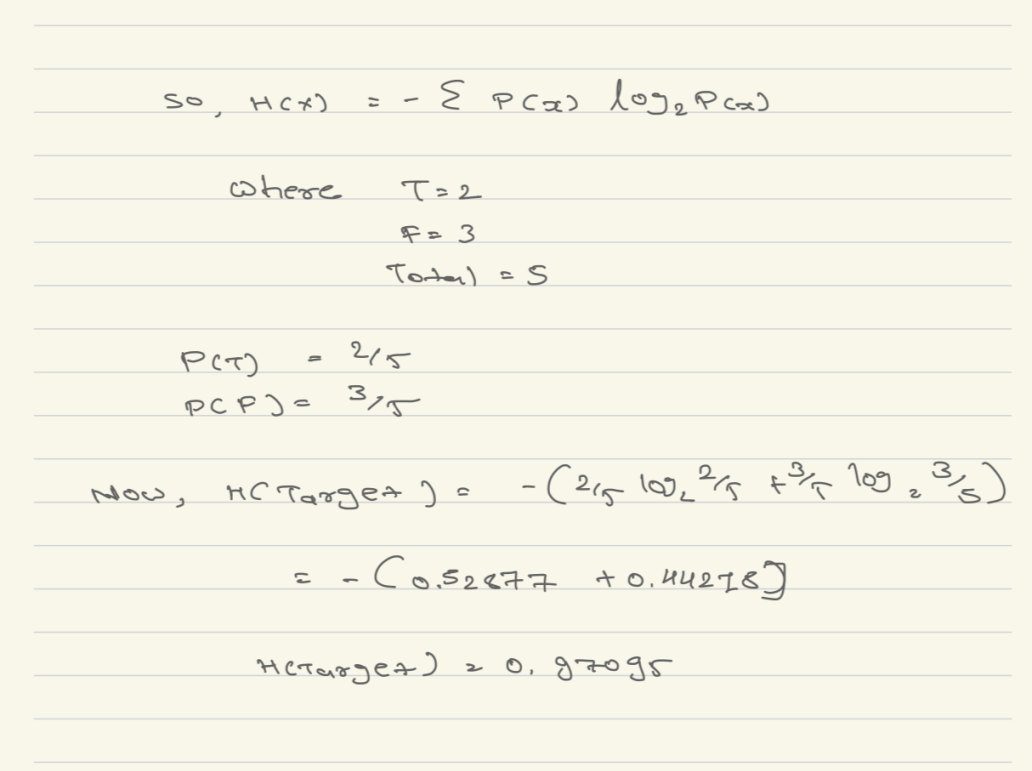
If the number of **hidden nodes** in the bottleneck layer exceeds the number of **input nodes**, the autoencoder risks learning the **identity function**, where it merely **copies the input to the output** without extracting meaningful features.

This can lead to **overfitting**, as the network has enough capacity to **memorize the training data** instead of learning compressed, generalized representations, reducing its ability to perform well on unseen data.

10. [3 pts/ea] Decision Tree
Given the following data

A	B	C	Target
Y	N	N	T
Y	N	Y	F
N	Y	N	F
Y	Y	Y	F
Y	Y	N	T

1) compute H(Target)



2) compute entropy (i.e., H(Target|A)), information gain and gain ratio of attribute A, respectively.

2) (i) H (Target | A)

1) subset 1: when, A = Y

∴ T = 2

∴ F = 2

∴ P(T | A = Y) = 2/4 = 0.5

∴ P(F | A = Y) = 2/4 = 0.5

(H Target | A = Y)

$= -2(0.5 \log_2 0.5)$

= 1

2) subset 2: when, A = N

T = 0

F = 2

H(Target | A = N) = 0

P (A = Y) = 4/5

P (A = N) = 1/5

H (Target | A) = (4/5 × 1) + (1/5 × 0)

$\therefore$ H (Target | A) = 0.8

Now information gain of A = ?

IG(A) = H(Target) - H(Target | A)

= 0.97095 - 0.8

Information gain (A) = 0.17095

Gain ratio of A = ?

$$H(A) = -(P(A=Y)\log_2 P(A=Y) + P(A=N)\log_2 P(A=N))$$

$$H(A) = -\left(\frac{4}{5}\log_2\frac{4}{5} + \frac{1}{5}\log_2\frac{1}{5}\right)$$

H(A) = -(0.25754 + 0.46439)

H(A) = 0.72193

$$\text{Gain Ratio (A)} = \frac{\text{Information Gain (A)}}{H(A)}$$

$$= \frac{0.17095}{0.72193}$$

Gain Ratio (A) = 0.2368

11. (coding) Linear Regression

1) complete the following program code of linear regression

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

```
# no_x: number of data
no_x=100
# generate 1D variable x and target y
x = np.linspace(0, 1, no_x)

# eps= level of random noise.
# You can change eps value, if you want
eps=0.2
y = 1 + x + eps * np.random.random(size=len(x))

# Refer to p. 19 in linear regression slides
# compute matrix X and vector y
'''
YOUR WORK HERE (2 pts)
# 1) change above x into a matrix X in p.19.
#     (add one column of '1's)
# 2) change y to a column vector
'''

# Refer to p. 20 in linear regression slides
# compute the coefficient of linear regression line
# draw data points and regression line
'''
YOUR WORK HERE (8 pts)
# compute (X^T X)^{-1} X^T y
# show two coefficient values
# suppose coeff is the coefficient values of above
# plot x, y and linear regression line together using plt
# may use plt.plot and plt.show
'''

# Make prediction
'''
YOUR WORK HERE (2 pts)
# Given a value of x=0.8, predict y value
'''
```

2) [bonus 4 pts] (Refer to p. 22 in the slides.) Implement linear regression using a library.

```
'''
YOUR WORK HERE
1) Implement linear regression using sklearn
   Use the same data as question 1)
2) Compare the results
'''
```

12. (coding) Logistic Regression

1) complete the following code of logistic regression

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification

# Hyperparameters
alpha = 0.01
# You can change num_iter value
num_iter = 1000

# Create dataset
X, y = make_classification(
      n_samples=100, n_features=2, n_redundant=0,
      n_informative=2, random_state=1, n_clusters_per_class=1 )

def add_ones(X):
  '''
  12-(A)
  YOUR WORK HERE (2 pts)
  # given a matrix X, add one column of '1's as the first column
  # you already did this in 11. 1)
  '''

# implement p_1(x; w) = 1/(1+exp(-w^T x))
def comp_p(X, w):
  '''
  12-(B)
  YOUR WORK HERE (2 pts)
  # given X and w, compute p_1(x; w)
  '''

X_1 = add_ones(X)

# initialize w=0
w = np.zeros(XX.shape[1])
m = y.size

for i in range(num_iter):
  '''
  12-(C)
  YOUR WORK HERE (8 pts)
  1) compute comp_p using X_1 and w
  # refer to p. 44
```

2) compute gradient $\sum_{i=1}^{N} x_{ij}[y_i - p_1(x_i; w)]\ /m$
(We use mean of gradients. Therefore, divide the original formula by m)
3) update w using gradient
‘’’

compute prediction probabilities using the coefficients w
‘’’
12-(D)
YOUR WORK HERE (3 pts)
after for loop is done:
Compute prediction prob p_1
Compute class value from p_1
Compute accuracy using X_1 and y (reuse X_1 as test data)
‘’’

2) Refer to p. 45-46 in slides.
Now we use regularization method in logistic regression to avoid overfitting problem.
Modify gradient using L2 term as follows

$$\frac{\partial}{\partial w_j}\ell(w) = \sum_{i=1}^{N} x_{ij}[y_i - p(x_i; w)]\ /m - \lambda w_i$$

And modify update formula of w_j as follows.

$$w_j = w_j + \alpha \sum_{i=1}^{N} x_{ij}[y_i - p(x_i; w)]\ /m - \alpha\lambda w_i$$

‘’’
YOUR WORK HERE (6 pts)
modify 12-(C) using L2 regularization term
repeat 12-(D)
show the difference between these two cases
‘’’

3) [bonus 6 pts] Implement using sklearn
‘’’
YOUR WORK HERE
Implement logistic regression using sklearn library.
Use the same data used in 12-1).
Compare the performance of sklearn version and your own implementation.
‘’’

13. (coding) Correlation analysis in Feature Selection

Implement the following code

```
import pandas as pd
import numpy as np
```

```
from sklearn.datasets import load_breast_cancer

# load breast cancer data
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = pd.Series(data.target)

‘’’
YOUR WORK HERE (4 pts)
# calculate the correlation matrix between the features and the target variable
# hint: use corr in pandas
# print the correlation coefficient of each feature
# sort the correlation values in descending order and select the top 10 features
 ‘’’
```

14. (coding) PCA

1) Complete the following program code

```
import numpy as np
import pandas as pd

# read iris dataset
data = pd.read_csv(‘iris.csv’)

# X: input features
X = data.iloc[:,0:4]

# Y: target
Y = data.iloc[:,4]

‘’’
YOUR WORK HERE (6 pts)
# compute covariance matrix of X (refer to p. 21)
# compute eigen values and eigen vector (hint: use np.linalg.eigh)
# choose TWO largest eigen values and their corresponding eigen vectors
# eigvec_set: two eigen vectors
 ‘’’
```

2) [4 pts] Visualization of PCA

```
# compute projected data pca_X (dimension reduced from 4 to 2)
pca_X = np.dot(eigvec_set.transpose() , X.transpose() ).transpose()

‘’’
YOUR WORK HERE
Show the pca_X in 2D graph
```

x axis: PC1, y axis: PC2
‘’’

3) [bonus 6 pts] Refer to p. 45 in slides.
‘’’
YOUR WORK HERE
Implement pca using sklearn and iris data.
Reduce dimensionality of iris data from 4 to 2.
Compare the results with question 14-1).
‘’’