NYCU Pattern Recognition, Homework 2

Deadline: Apr. 05, 23:59

Part. 1, Coding (70%):

In this coding assignment, you are requested to implement 1) logistic Regression and 2) Fisher's Linear Discriminant by using only Numpy, then train your model on the provided dataset and finally evaluate the performance on testing data. Please train your logistic regression model using Gradient Descent, not the closed-form solution.

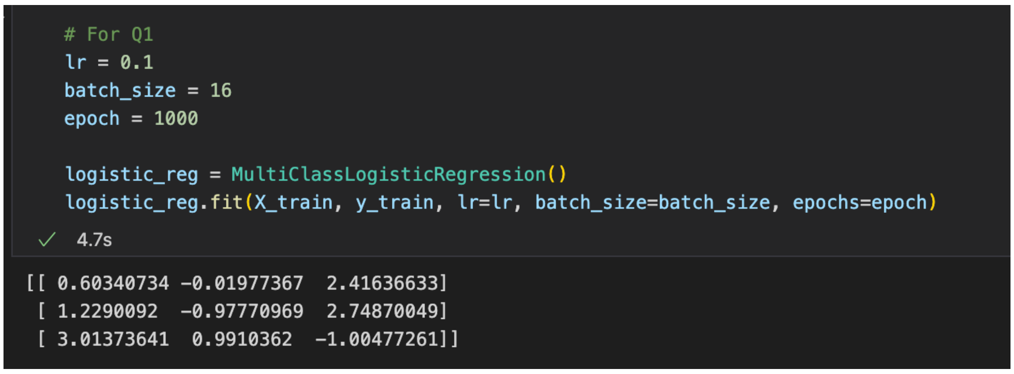
(20%) Logistic Regression Model

Requirements:

- Use Gradient Descent
- Use CE (Cross-Entropy) as your loss function.
- Use Softmax for this multiclass classification task.

Criteria:

1. (0%) Show the learning rate, epoch, and batch size that you used.

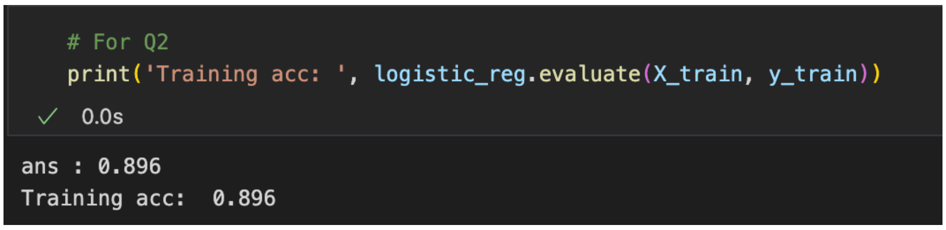


```
# For Q1
lr = 0.1
batch_size = 16
epoch = 1000

logistic_reg = MultiClassLogisticRegression()
logistic_reg.fit(X_train, y_train, lr=lr, batch_size=batch_size, epochs=epoch)
```

✓ 4.7s

```
[[ 0.60340734 -0.01977367  2.41636633]
 [ 1.2290092  -0.97770969  2.74870049]
 [ 3.01373641  0.9910362  -1.00477261]]
```

2. (5%) What's your training accuracy?

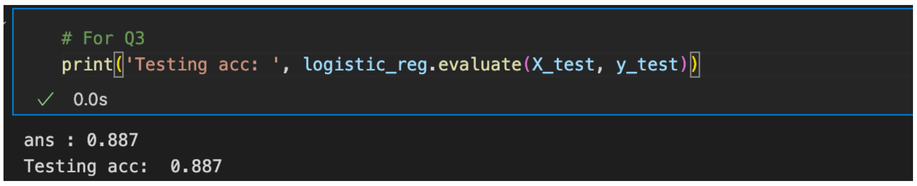


```
# For Q2
print('Training acc: ', logistic_reg.evaluate(X_train, y_train))
```

✓ 0.0s

```
ans : 0.896
Training acc:  0.896
```

3. (5%) What's your testing accuracy?

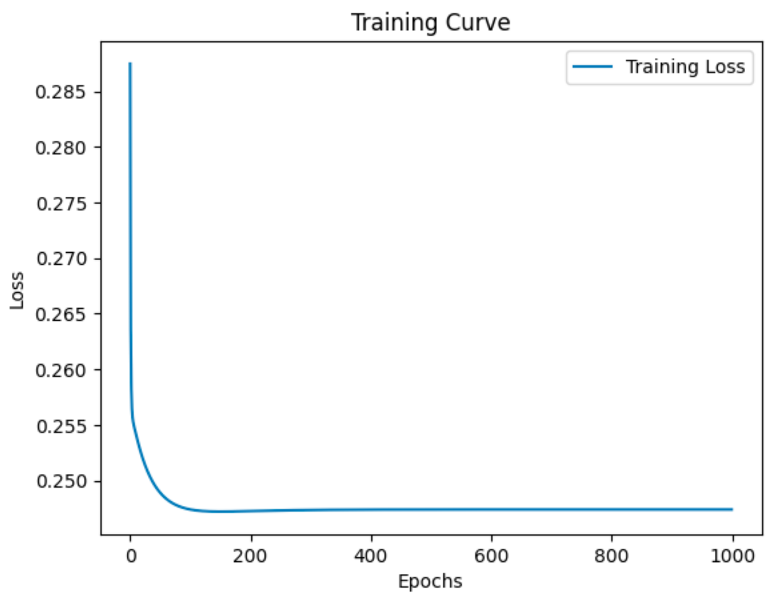


```
# For Q3
print('Testing acc: ', logistic_reg.evaluate(X_test, y_test))
```

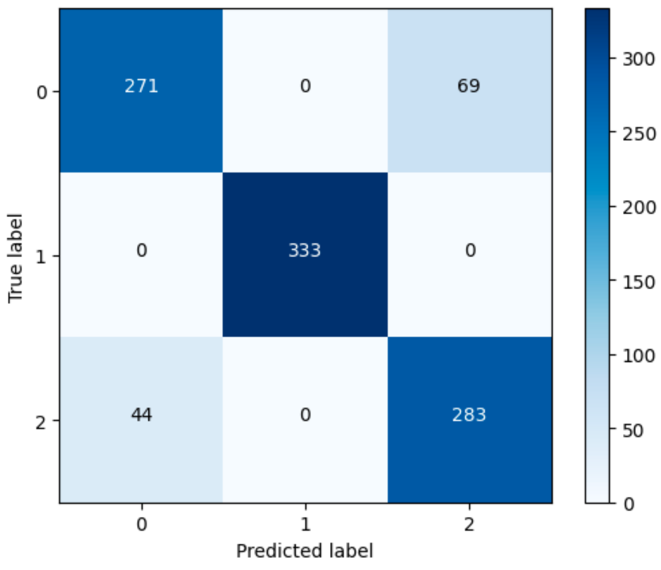
✓ 0.0s

```
ans : 0.887
Testing acc:  0.887
```

4. (5%) Plot the learning curve of the training. (x-axis=epoch, y-axis=loss)



5. (5%) Show the confusion matrix on testing data.



	0	1	2
0	271	0	69
1	0	333	0
2	44	0	283

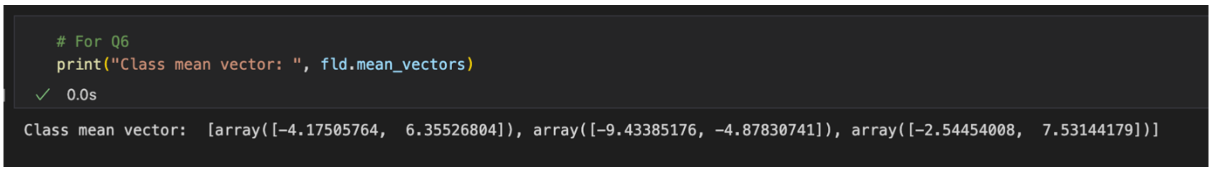
(30%) Fisher' s Linear Discriminant (FLD) Model

Requirements:

- Use FLD to reduce the dimension of the data from 2 to 1.

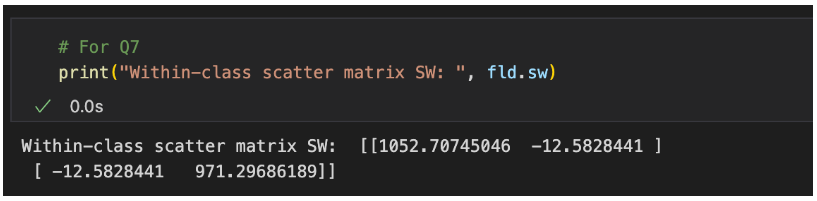
Criteria:

6. (2%) Compute the mean vectors m_i (i=1, 2, 3) of each class on training data.



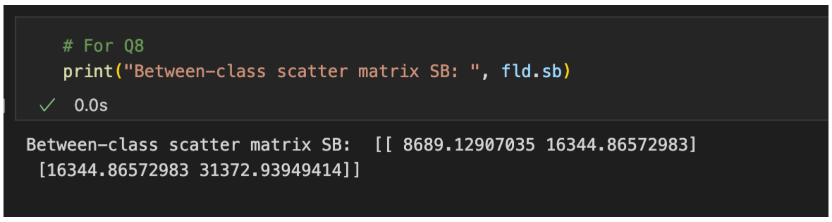
```
# For Q6
print("Class mean vector: ", fld.mean_vectors)
✓ 0.0s

Class mean vector:  [array([-4.17505764,  6.35526804]), array([-9.43385176, -4.87830741]), array([-2.54454008,  7.53144179])]
```

7. (2%) Compute the within-class scatter matrix S_W on training data.



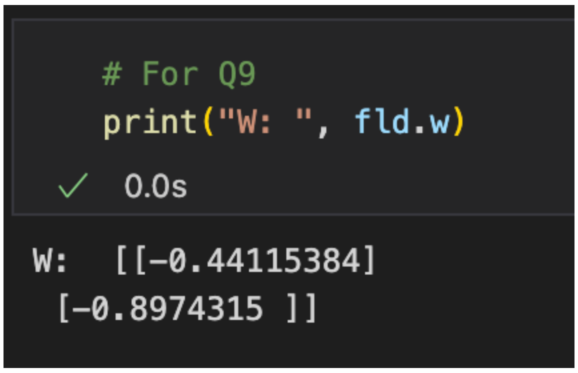
```
# For Q7
print("Within-class scatter matrix SW: ", fld.sw)
✓ 0.0s
Within-class scatter matrix SW:  [[1052.70745046  -12.5828441 ]
 [ -12.5828441   971.29686189]]
```

8. (2%) Compute the between-class scatter matrix S_B on training data.



```
# For Q8
print("Between-class scatter matrix SB: ", fld.sb)
✓ 0.0s
Between-class scatter matrix SB:  [[ 8689.12907035 16344.86572983]
 [16344.86572983 31372.93949414]]
```

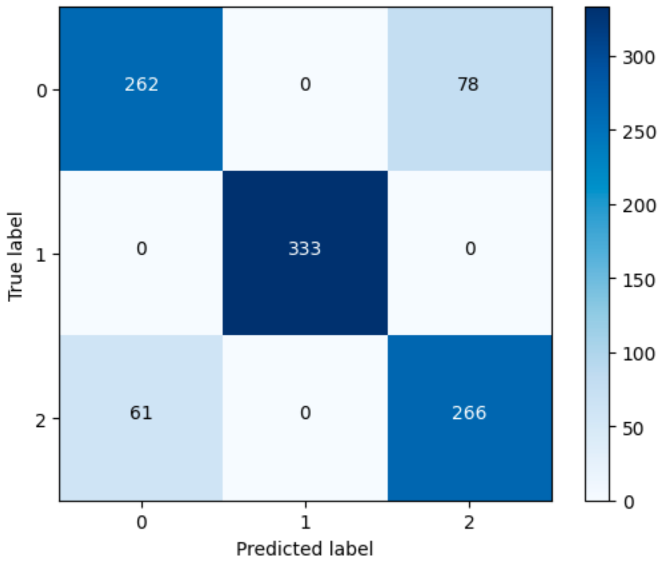
9. (4%) Compute the Fisher’ s linear discriminant w on training data.



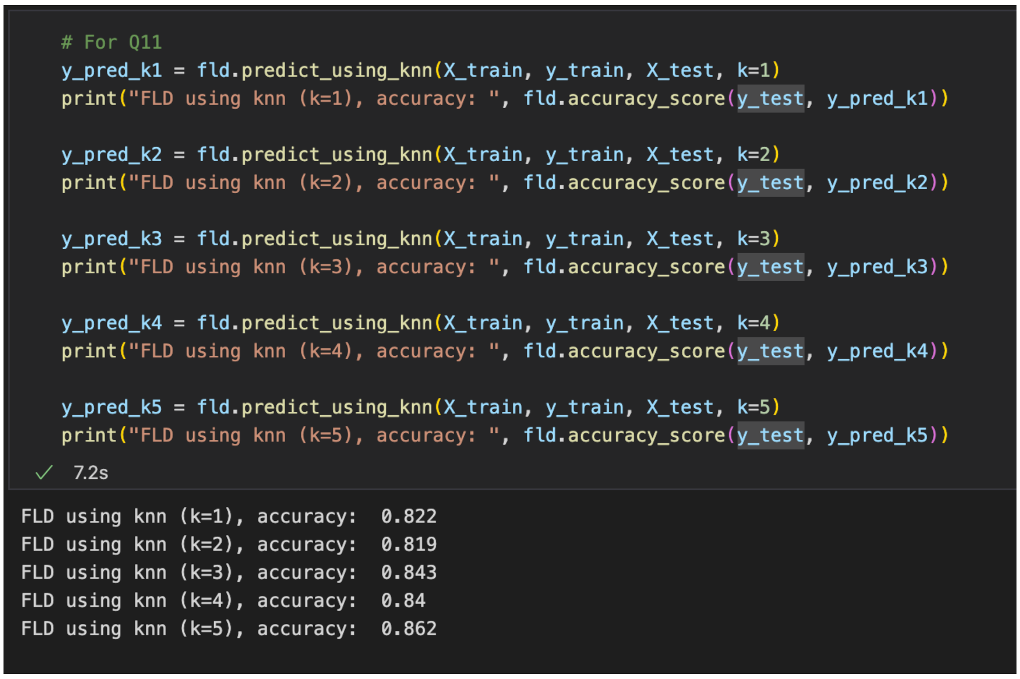
```
# For Q9
print("W: ", fld.w)
✓ 0.0s
W:  [[-0.44115384]
 [-0.8974315 ]]
```

10. (8%) Project the testing data to get the prediction using the shortest distance to the class mean. Report the accuracy score and draw the confusion matrix on testing data.

FLD using class mean, accuracy: 0.861



11. (8%) Project the testing data to get the prediction using K-Nearest-Neighbor. Compare the accuracy score on the testing data with K values from 1 to 5.



```
# For Q11
y_pred_k1 = fld.predict_using_knn(X_train, y_train, X_test, k=1)
print("FLD using knn (k=1), accuracy: ", fld.accuracy_score(y_test, y_pred_k1))

y_pred_k2 = fld.predict_using_knn(X_train, y_train, X_test, k=2)
print("FLD using knn (k=2), accuracy: ", fld.accuracy_score(y_test, y_pred_k2))

y_pred_k3 = fld.predict_using_knn(X_train, y_train, X_test, k=3)
print("FLD using knn (k=3), accuracy: ", fld.accuracy_score(y_test, y_pred_k3))

y_pred_k4 = fld.predict_using_knn(X_train, y_train, X_test, k=4)
print("FLD using knn (k=4), accuracy: ", fld.accuracy_score(y_test, y_pred_k4))

y_pred_k5 = fld.predict_using_knn(X_train, y_train, X_test, k=5)
print("FLD using knn (k=5), accuracy: ", fld.accuracy_score(y_test, y_pred_k5))
```

✓ 7.2s

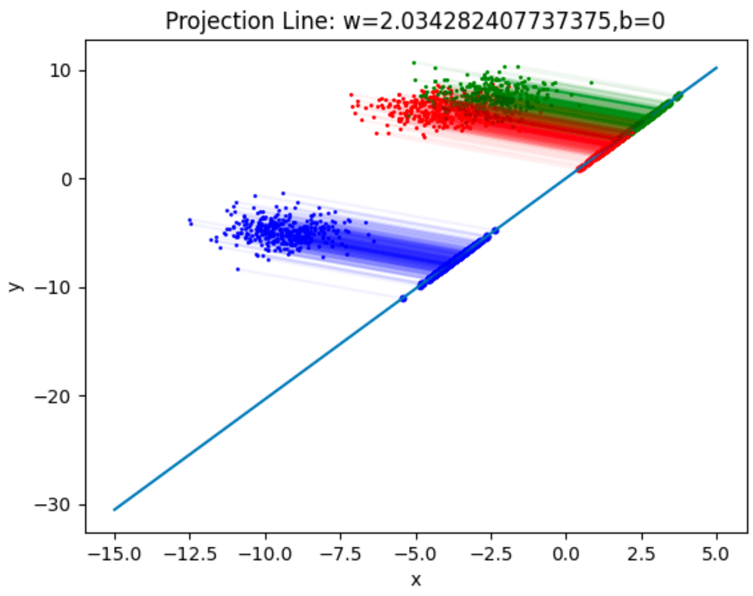
```
FLD using knn (k=1), accuracy:  0.822
FLD using knn (k=2), accuracy:  0.819
FLD using knn (k=3), accuracy:  0.843
FLD using knn (k=4), accuracy:  0.84
FLD using knn (k=5), accuracy:  0.862
```

12. (4 %)

1) Plot the best projection line on the training data and show the slope and intercept on the title *(you can choose any value of intercept for better visualization)*

2) colorize the training data with each class

3) project all training data points on your projection line. Your result should look like the b elow image (This image is for reference, not the answer)



(20%) Train your own model

Requirements:

- Using another dataset that we provided (a real-world dataset).
- Train your model (FLD or Logistics Regression model that you implemented above).

- Try different parameters and feature engineering to beat the baseline.
- Save your testing predictions in the CSV file.

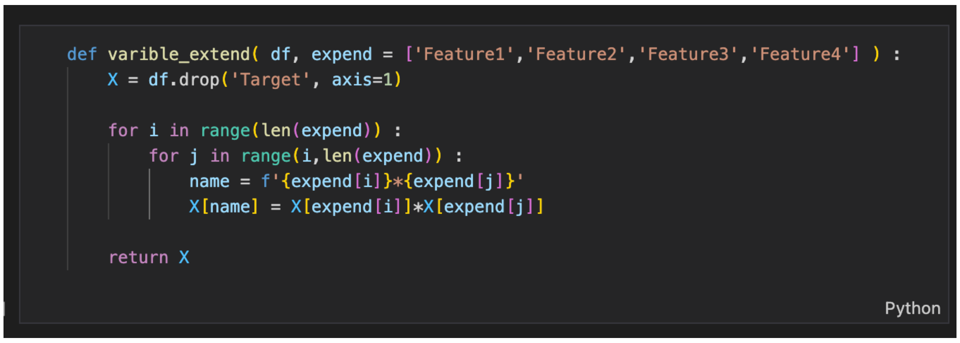
Criteria:

13. Explain how you chose your model and what feature processing you have done in detail. Otherwise, no points will be given.

Point	Accuracy
20	testing acc > 0.921
15	0.91 < testing acc <= 0.921
8	0.9 < testing acc <= 0.91
0	testing acc <= 0.9

batch_size: 8
learning rate : 0.9
epoch : 10000000

Use feature :



```python
def varible_extend( df, expend = ['Feature1','Feature2','Feature3','Feature4'] ) :
    X = df.drop('Target', axis=1)

    for i in range(len(expend)) :
        for j in range(i,len(expend)) :
            name = f'{expend[i]}*{expend[j]}'
            X[name] = X[expend[i]]*X[expend[j]]

    return X
```

I used logistic regression as my classifier and did feature expansion because I did not know the effect of feature on this Target, so I left all the expansion features, which can be learned from the last homework that more features are more helpful to the regression performance.

Part. 2, Questions (30%):

(6%) 1. Discuss and analyze the performance

a) between <u>Q10</u> and <u>Q11</u>, which approach is more suitable for this dataset. Why?

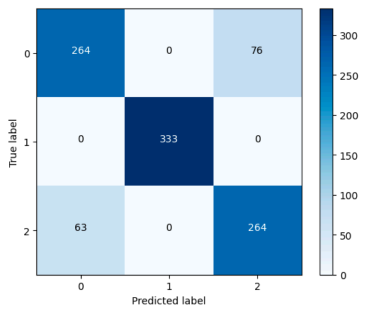
b) between different values of k in Q11. (Which is better, a larger or smaller k? Does this always hold?)

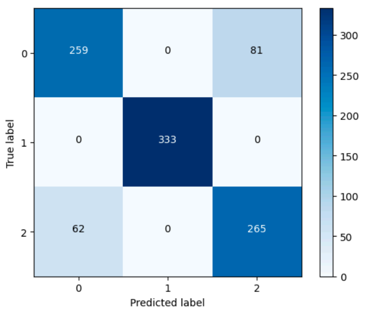
(a) According to Q12, it can be found that two classes are very close to each other, so when using knn, k is set a little larger, it is easy to be biased towards a certain class, but it is not necessarily correct, we can learn from Q11, when k=4 the accuracy rate will be lower than k=3, in this data set should take the class mean to predict better

(b)

k	4	5	10	20	100	327	all
accuracy	0.84	0.862	0.854	0.861	0.854	0.857	0.327

The number of this dataset class is 339 334 337, so I tried the numbers in the table above to experiment and found that the larger the number, the better the result.





k=20 : k=327

Because this time the dataset has two classes overlapping each other, so when the k is set a little too large, but will be estimated wrong

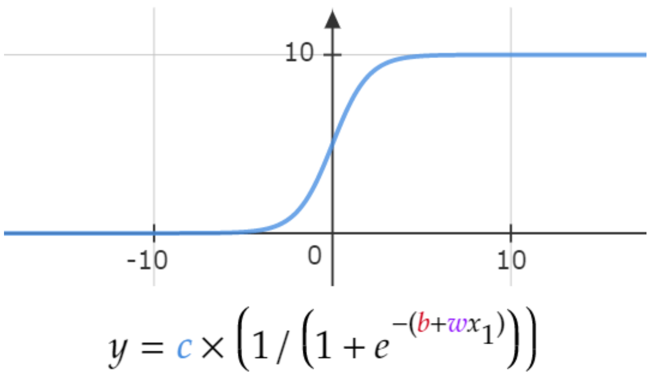
(6%) 2. Compare the sigmoid function and softmax function.

(1). Range of values:

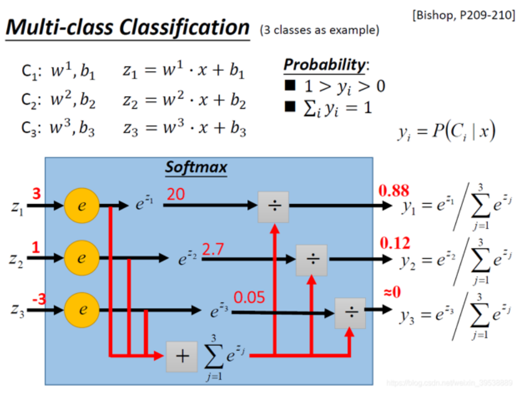
The sigmoid function maps any real-valued number to a value between 0 and 1. This makes it suitable for binary classification problems where the output is either 0 or 1. In contrast, the softmax function maps a vector of real numbers to a probability distribution, meaning that the output values are between 0 and 1 and sum up to 1. This makes it suitable for multiclass classification problems.

(2). Shape of the curve:

The sigmoid function has an S-shaped curve and is monotonically increasing, which means that the output increases as the input increases. In contrast, the softmax function has a curved shape, and the output is not necessarily monotonically increasing.



sigmoid


Multi-class Classification (3 classes as example)
[Bishop, P209-210]
C_1: w^1, b_1 $z_1 = w^1 \cdot x + b_1$
C_2: w^2, b_2 $z_2 = w^2 \cdot x + b_2$
C_3: w^3, b_3 $z_3 = w^3 \cdot x + b_3$
Probability:
$1 > y_i > 0$
$\sum_i y_i = 1$
$y_i = P(C_i \mid x)$
Softmax
3
1
-3
20
2.7
0.05
0.88
0.12
≈0
$y_1 = e^{z_1} / \sum_{j=1}^{3} e^{z_j}$
$y_2 = e^{z_2} / \sum_{j=1}^{3} e^{z_j}$
$y_3 = e^{z_3} / \sum_{j=1}^{3} e^{z_j}$

softmax

(6%) 3. Why do we use cross entropy for classification tasks and mean square error for regression tasks?

Cross entropy for classification tasks:

Cross entropy is a loss function that measures the difference between the predicted probability dist ribution and the actual probability distribution for a classification problem. In other words, it meas ures the distance between the predicted class probabilities and the true class probabilities. Cross en tropy is used in classification tasks because it is more suited to handling probability distributions a nd penalizes more strongly for larger prediction errors.

Mean square error for regression tasks:

Mean square error is a loss function that measures the average squared difference between the pre dicted values and the true values. It is used in regression tasks because it is well-suited for continu ous and numeric data. In regression tasks, the goal is to predict a continuous value, such as a real number, and mean square error provides a measure of the average squared difference between the predicted values and the true values.

In summary, cross entropy is more suited to classification tasks because it can handle probability d istributions and penalizes more strongly for larger prediction errors.

(6%) 4. In Q13, we provide an imbalanced dataset. Are there any methods to improve Fisher Linea r Discriminant's performance in handling such datasets?

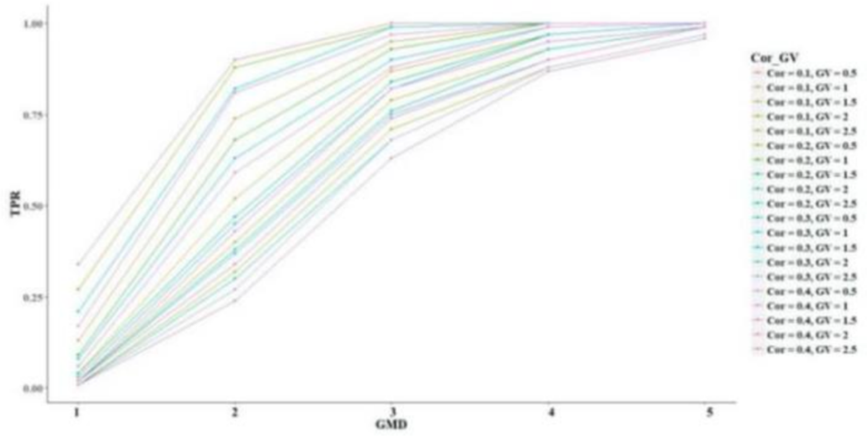
Yes! Here are a few ways to solve the problem

Sampling Techniques :

Sampling techniques are a group of methods used to address imbalanced datasets in machine learn ing. These techniques aim to balance the distribution of the classes in the dataset by either undersa mpling the majority class or oversampling the minority class.

According to "provide an imbalanced dataset. are there any methods to improve Fisher Linear Dis criminant's performance in handling such datasets? "The authors of this paper propose three metho ds to solve this problem

LDA :



(c) LDA

Random Undersampling: (RUS)

This technique involves randomly selecting a subset of examples from the majority class to balan ce the class distribution. While it is simple to implement, it can lead to loss of important informati on as some examples from the majority class may be discarded.

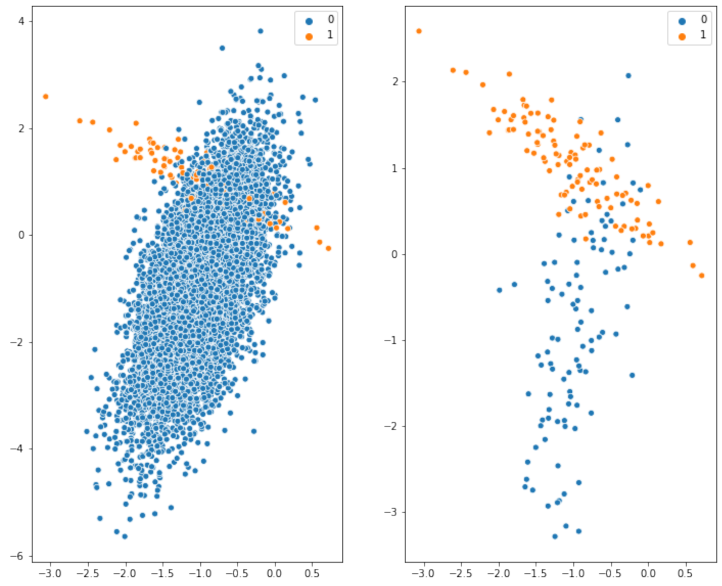
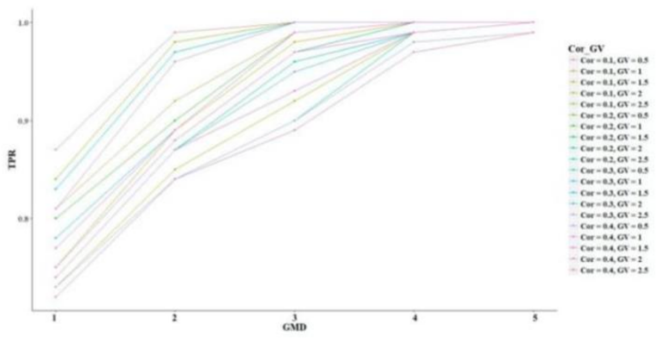


Figure : Training data (left), Undersampling (right)



(b) RUS-LDA

Random Oversampling: (ROS)

This technique involves randomly replicating examples from the minority class until the class distr ibution is balanced. While this method is easy to implement and can prevent loss of information, it can also lead to overfitting and increased computational costs.

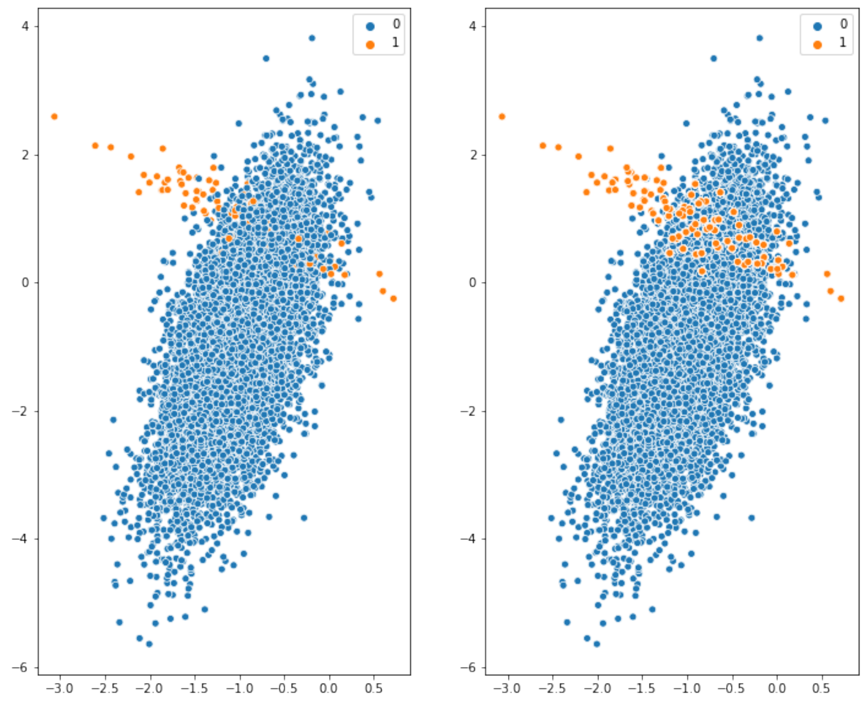
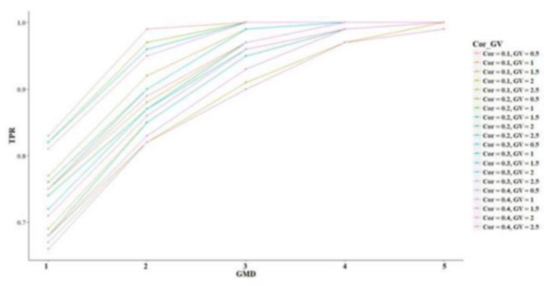


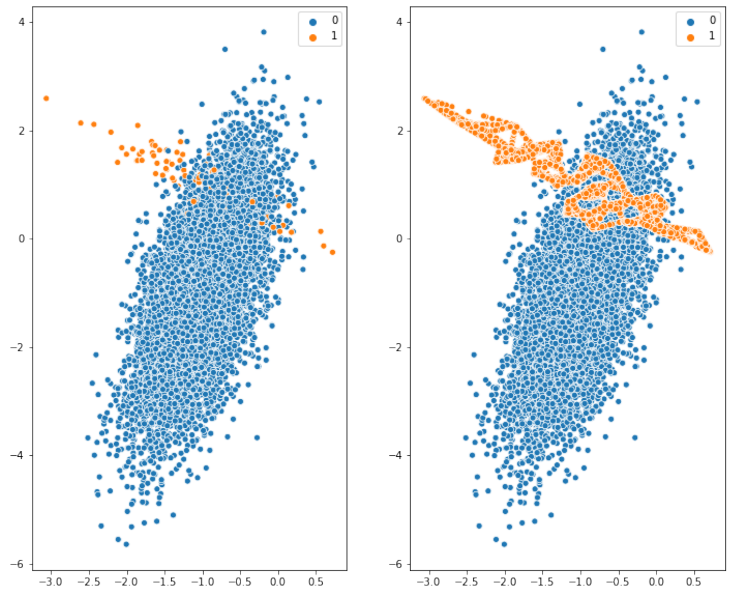
Figure : Training data (left), Oversampling (right)



(a) ROS-LDA

Synthetic Minority Over-sampling Technique (SMOTE):

SMOTE is an oversampling technique that generates synthetic examples for the minority class by interpolating between existing examples. It works by selecting a minority class example and finding its k-nearest neighbors. It then creates synthetic examples by randomly interpolating between the selected example and its neighbors. SMOTE can help to prevent overfitting and can lead to improved generalization.

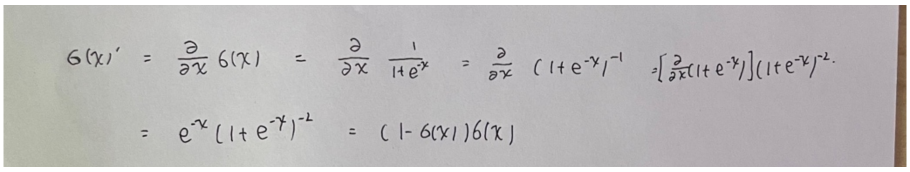


(6%) 5. Calculate the results of the partial derivatives for the following equations. (The first one is binary cross-entropy loss, and the second one is mean square error loss followed by a sigmoid function.)

$$\frac{\partial}{\partial x}\left(y * \ln\left(\sigma(x)\right) + (1-y) * \ln\left(1-\sigma(x)\right)\right)$$

$$\frac{\partial}{\partial x}\left(\left(y-\sigma(x)\right)^2\right)$$

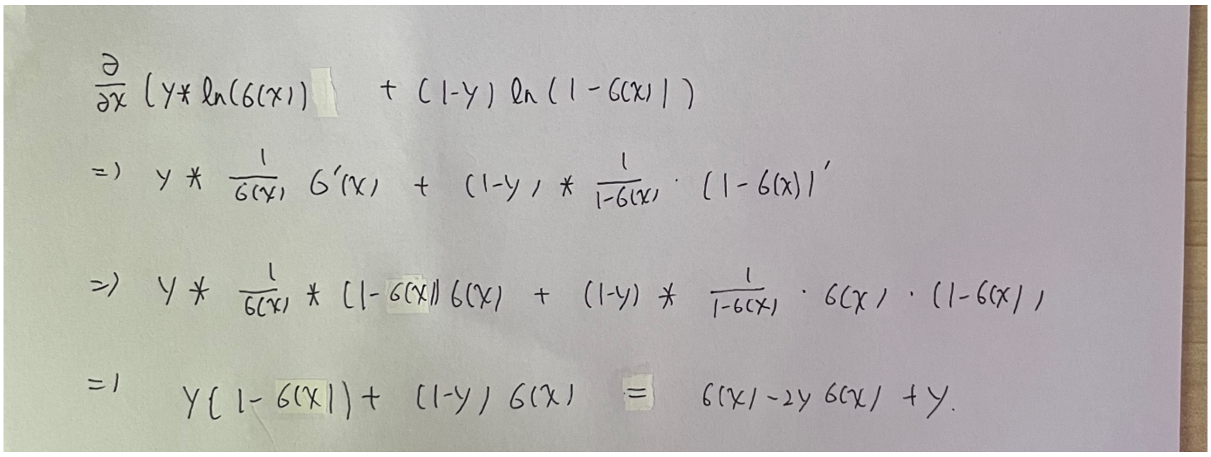
sigmoid derivatives



$$\sigma(x)' = \frac{\partial}{\partial x}\sigma(x) = \frac{\partial}{\partial x}\frac{1}{1+e^{-x}} = \frac{\partial}{\partial x}(1+e^{-x})^{-1} = -\left[\frac{\partial}{\partial x}(1+e^{-x})\right](1+e^{-x})^{-2}.$$

$$= e^{-x}(1+e^{-x})^{-2} = (1-\sigma(x))\sigma(x)$$

(1)



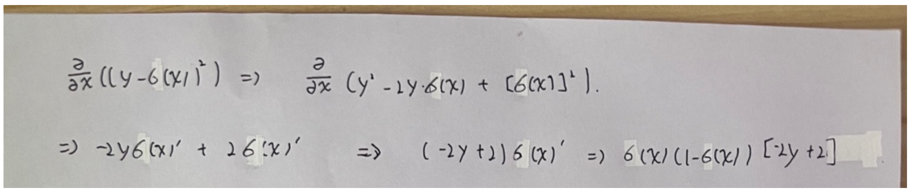
$$\frac{\partial}{\partial x}(y*\ln(\sigma(x)) + (1-y)\ln(1-\sigma(x)))$$

$$\Rightarrow y * \frac{1}{\sigma(x)} \sigma'(x) + (1-y) * \frac{1}{1-\sigma(x)} \cdot (1-\sigma(x))'$$

$$\Rightarrow y * \frac{1}{\sigma(x)} * (1-\sigma(x))\sigma(x) + (1-y) * \frac{1}{1-\sigma(x)} \cdot \sigma(x) \cdot (1-\sigma(x))$$

$$\Rightarrow y(1-\sigma(x)) + (1-y)\sigma(x) = \sigma(x) - 2y\sigma(x) + y.$$

(2)



$$\frac{\partial}{\partial x}((y-\sigma(x))^2) \Rightarrow \frac{\partial}{\partial x}(y^2 - 2y\sigma(x) + [\sigma(x)]^2).$$

$$\Rightarrow -2y\sigma(x)' + 2\sigma(x)' \Rightarrow (-2y+2)\sigma(x)' \Rightarrow \sigma(x)(1-\sigma(x))[-2y+2]$$