Metrics

In a population where the amount of cats is equal to the amount of dogs. Considering the following classification results from a classifier.

Model A	Predicted dog	Predicted cat
Actual dog	30	20
Actual cat	10	40

T1. What is the accuracy of Model A?

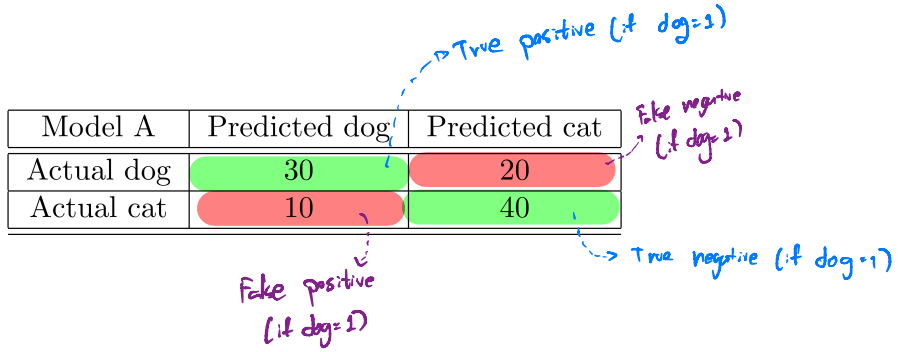
T2. Consider cats as 'class 1' (positive) and dogs as 'class 0' (negative), calculate the precision, recall, and F1.

T3. Consider class cat as 'class 0' and class dog as 'class 1', calculate the precision, recall, and F1.

It is important to specify the 'positive' class when you calculate precision, recall, and F1. If there are more than two classes, it is usually done in a one-versus-all setting where one class is considered positive and the rest of the classes are considered negative.

T4. Now consider a lopsided population where there are 80% cats. What is the accuracy of Model A? Using dog as the positive class, what is the precision, recall, and F1? Explain how and why these numbers change (or does not change) from the previous questions.

ANS T1.



Thus, the accuracy is $\frac{TP+TN}{TP+TN+FP+FN} = \frac{30+40}{30+40+20+10} = \frac{7}{10}$ or 70%

ANS T2.

If cat is `class 1` and dog is `class 2`

	Model A	Predicted dog	Predicted cat
(0)	Actual dog	30 TN	20 FP
(1)	Actual cat	10 FN	40 TP

$\text{Precision} = \frac{TP}{TP+FP} = \frac{40}{40+20} = \frac{2}{3}$ or about 67%

$\text{Recall} = \frac{TP}{TP+FN} = \frac{40}{40+10} = \frac{4}{5}$ or about 80%

$F1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{1}{\frac{1}{(\frac{2}{3})} + \frac{1}{(\frac{4}{5})}} = \frac{8}{11}$ or about 73%

ANS T3.

If cat is "class 0" and dog is "class 1"

Model A	Predicted dog	Predicted cat
(1) Actual dog	30 TP	20 FN
(0) Actual cat	10 FP	40 TN

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{30}{30+20} = \frac{3}{5} \text{ or about } 60\%$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{30}{30+10} = \frac{3}{4} \text{ or about } 75\%$$

$$F1 = 2 \cdot \frac{1}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} = 2 \cdot \frac{1}{\frac{1}{\frac{3}{5}} + \frac{1}{\frac{3}{4}}} = \frac{2}{3} \text{ or about } 67\%$$

ANS T4.

20
80

If the population of cat is 80% of all population

Also, we label dog as "class 1" (Positive class)

50%, 50%: total$_{dog}$ = 50, total$_{cat}$ = 50

Model A	Predicted dog	Predicted cat
Actual dog	30	20
Actual cat	10	40

→

Model A	Predicted dog	Predicted cat
(1) Actual dog	0.2×30 = 6 = 2 = 12	0.2×20 = 4 = 2 = 8
(0) Actual cat	0.8×10 = 8 = 2 = 16	0.8×40 = 32 = 2 = 64

total$_{dog}$ = 10 : 2 = 20
total$_{cat}$ = 40 : 2 = 80

Since we time dog's 0.2

This make population size the same.

Thus the new accuracy will be : $\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{12+64}{12+64+16+8} = \frac{19}{25}$ or about 76%

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{12}{12+8} = \frac{3}{5} \text{ or about } 60\%$$

Same as before

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{12}{12+16} = \frac{3}{7} \text{ or about } 43\%$$

Decreased!

$$F1 = 2 \cdot \frac{1}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} = 2 \cdot \frac{1}{\frac{1}{\frac{3}{5}} + \frac{1}{\frac{3}{7}}} = \frac{1}{2} \text{ or about } 50\%$$

Decreased!!

Reason:

The accuracy of this data is increased because the errors or False Positive and False Negative are scaled down especially False Negative(Dog). In the same way for Precision and F1, the amount of True Positive is decreased and the False Positive is scaled up. Also for F1 which got weighting effects from Precision (decreased!) even though the accuracy is increased but it has nothing to do with True Negative.

ANS for OT1

To see when accuracy = F1, we can set the equation like in the following:

$$\text{Accuracy} = F1$$

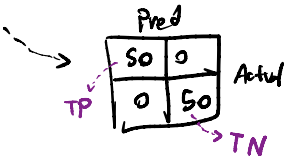
$$\frac{TP+TN}{TP+TN+FP+FN} = \frac{2TP}{2TP+FP+FN}$$

Then we can come up w/ some scenario that make this equation equal!

For the perfect scenario where both are equal to 1, (No error!)

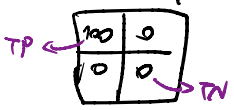
$$\frac{TP+TN}{TP+TN+FP+FN} = \frac{2TP}{2TP+FP+FN}$$

FP + FN → equal to "0"



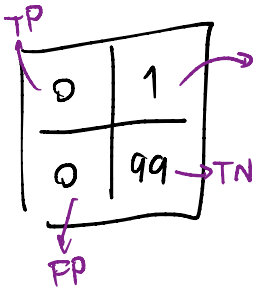
or

very bias positive



∴ F1 = Accuracy

For the case when Accuracy > F1 is when we have biased data like:



This example has accuracy of 99% but F1 of 0%

Because, $\frac{TP+TN}{TP+TN+FP+FN} = \frac{2TP}{2TP+FP+FN}$

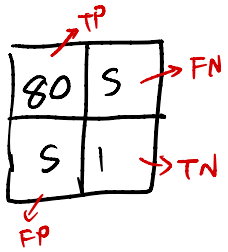
"Accuracy" accounted the "TN" where "F1" doesn't have!

For the case when "Accuracy is less than F1",

we can analyze it opposite way as the case where "Accuracy > F1":

Since we know "Accuracy rely on TN but F1 don't"

Then if we have the following cases:



19%

$$\frac{TP+TN}{TP+TN+FP+FN} = \frac{2TP}{2TP+FP+FN}$$

Eval: Accuracy is 87% and F1 is 94%

To conclude:

The accuracy and F1 will be equal when:

1. No errors
2. True positive >= 1 and others are zero.

The accuracy > F1 when:

→ The True positive has lower amount compared to True negative and there are errors in False Negative/Positive more than True Positive.

The accuracy < F1 when:

→ The True Positive has higher amount than True negative and the Σ of errors should not greater than True Positive.

[Optional] Fun with matrix algebra

Prove the following statements. All of them can be solved by first expanding out the matrix notation as a combination of their elements, and then use the definitions of trace and matrix derivatives to help finish the proof. For example, the (i, j) element of $Y = AB$ is $Y_{i,j} = \Sigma_m A_{i,m} B_{m,j}$.

OT5. $\nabla_A trAB = B^T$

OT6. $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$

OT7. $\nabla_A trABA^TC = CAB + C^TAB^T$

Hint: Try first solving the easier equation of $\nabla_A trBAC = (CB)^T = B^TC^T$

ANS for OT5

Let $Y = AB$

and each element of Y (i,j) is $Y_{ij} = \sum_m^N A_{im} B_{mj}$, where m start from 1 to N

then, $tr\,AB = tr\,Y = \sum_i^N Y_{ii}$

$$tr\,Y = \sum_i^N \left(\sum_m^N A_{im} B_{mi} \right)$$

Do derivative w.r.t A,

$$\nabla_A tr\,AB = \nabla_A tr\,Y = \sum_i^N \left(\sum_m^N B_{mi} \right)$$, A terms are dominated

$$\nabla_A tr\,AB = \nabla_A tr\,Y = \sum_i^N \left(B_{1i} + B_{2i} + B_{3i} + \ldots + B_{mi} \right)$$

$$\therefore \nabla_A tr\,AB = \sum_{i,m}^N B_{m,i}$$, where $m = i \rightarrow$ equals to $\sum_i^N B_{i,i}$

since $tr\,B = tr\,B^T$ then $\nabla_A tr\,AB = B^T$

ANS for OTG

Let A be a matrix $n \times m$ then,

$$\therefore \quad \nabla_{A^T} f(A) = \begin{bmatrix} \frac{\partial f(A)}{\partial A_{11}} & \frac{\partial f(A)}{\partial A_{21}} & \cdots & \frac{\partial f(A)}{\partial A_{n1}} \\ \frac{\partial f(A)}{\partial A_{12}} & & & \vdots \\ \vdots & & & \frac{\partial f(A)}{\partial A_{nm}} \\ \frac{\partial f(A)}{\partial A_{1m}} & \cdots & & \end{bmatrix} = (\nabla_A f(A))^T$$

T5. If the starting points are (3,3), (2,2), and (-3,-3). Describe each assign and update step. What are the points assigned? What are the updated centroids? You may do this calculation by hand or write a program to do it.

ANSWER

I did the calculation by writing a program to compute the results and plot it as shown in Figure 1.

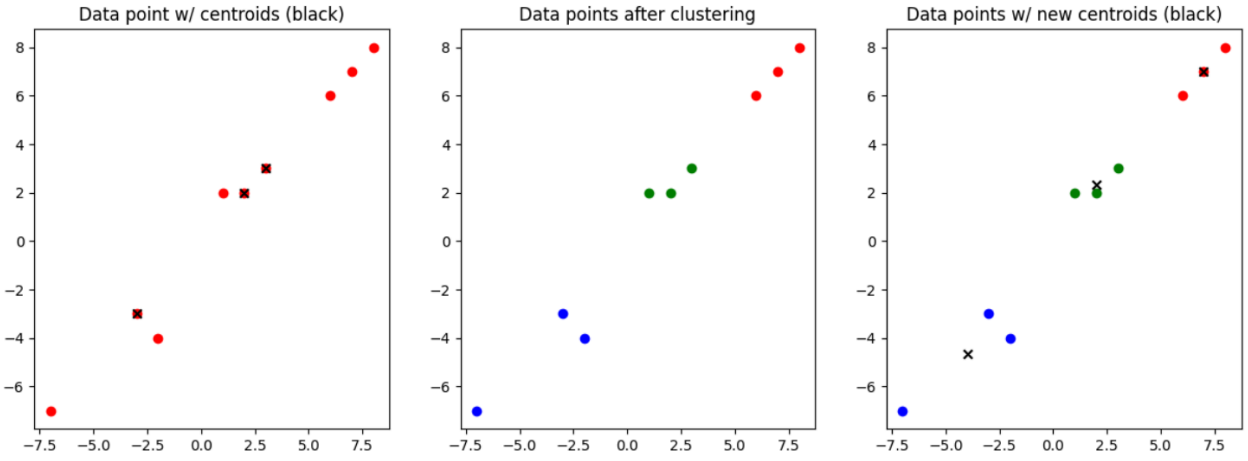


Figure 1: The plot of original data points in red color with their defined centroids in black color (left). After clustering, the K-means algorithm clearly distinguishes the data points into three groups (middle). The rightmost plot shows the updated centroids in black color.

For the value of updated centroids are: (7, 7), (2, 2.33), (-4, -4.66)

The Figure 2 shows the step for both assigning and updating for **T5**.

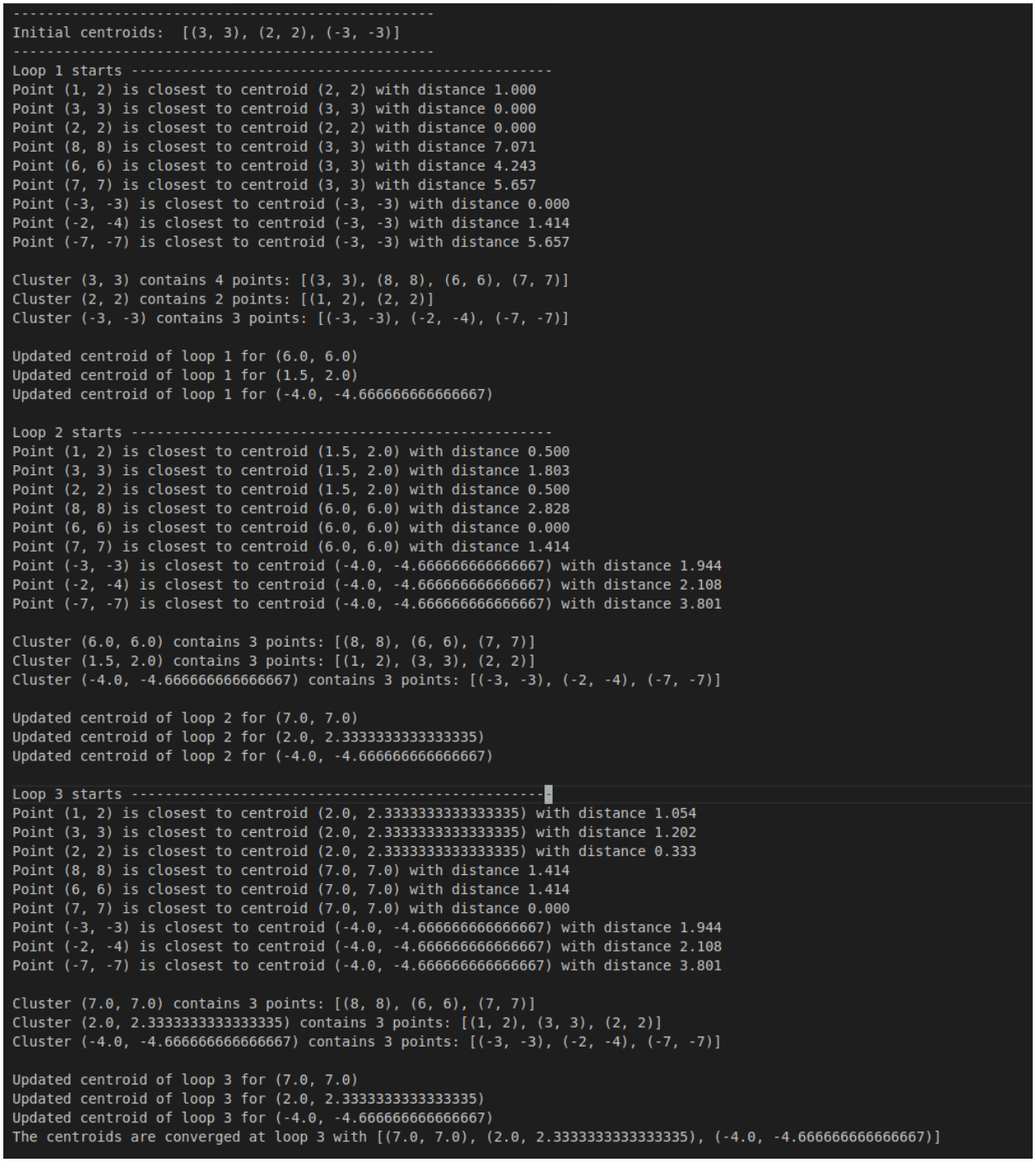


Figure 2: The result in the terminal shows how each data point is assigned and what the new centroids (mean) of each cluster after assigning all the points. Also, there is the final result of how each data point is in each cluster with its new centroids.

T6. If the starting points are (-3,-3), (2,2), and (-7,-7), what happens?

ANSWER

After switching the starting points, the K-means gives the totally different result as shown in Figure 3.

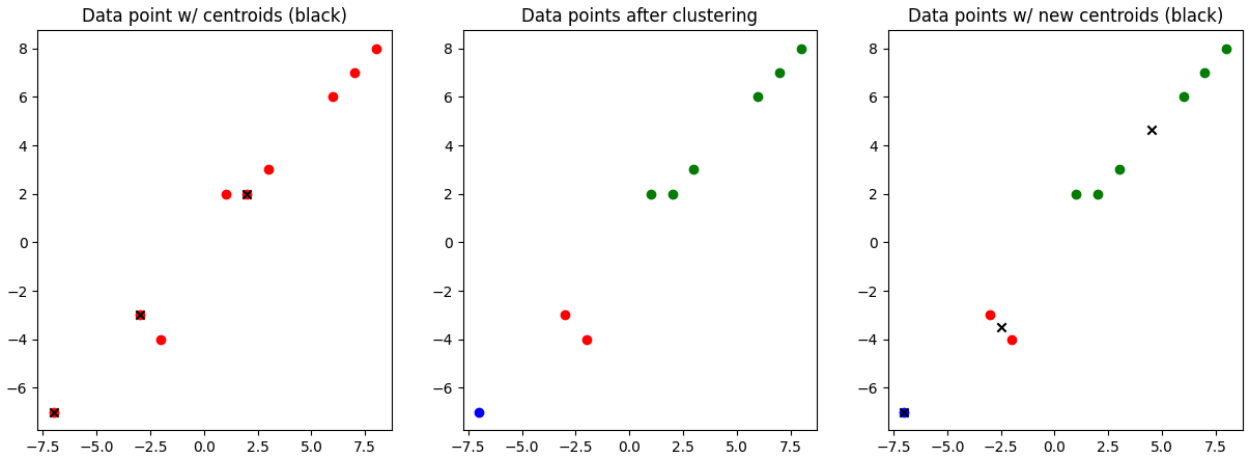


Figure 3: The plot (after switching starting points) of original data points in red color with their defined centroids in black color (left). After clustering, the K-means algorithm clearly distinguishes the data points into three groups (middle). The rightmost plot shows the updated centroids in black color.

For the value of updated centroids are: (-2.5, -3.5), (4.5, 4.66), (-7, -7)

The Figure 4 shows the step for both assigning and updating for **T6**.

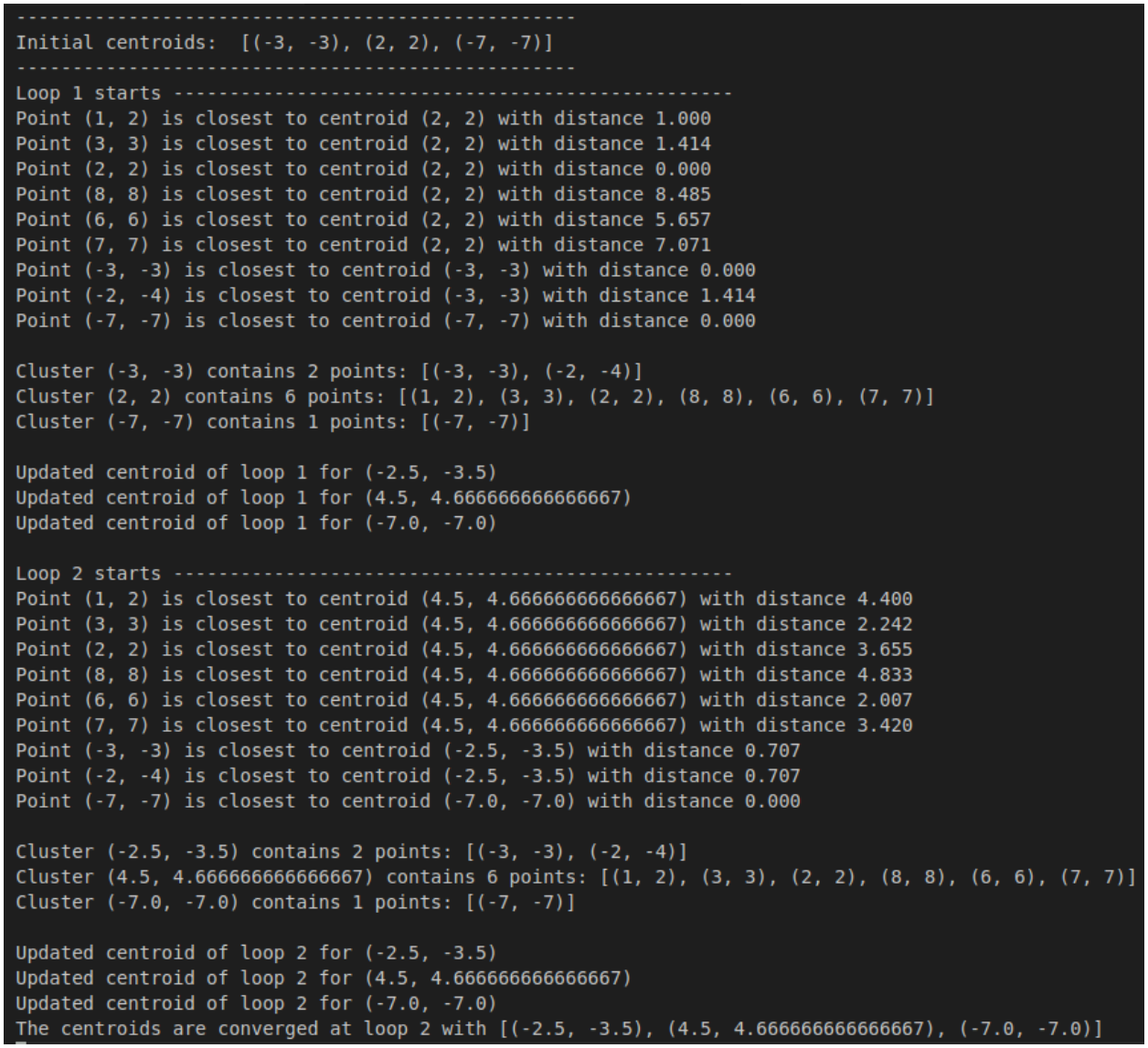


Figure 4: The result in the terminal shows how each data point is assigned and what the new centroids (mean) of each cluster after assigning all the points. Also, there is the final result of how each data point is in each cluster with its new centroids.

T7. Between the two starting set of points in the previous two questions, which one do you think is better? How would you measure the 'goodness' quality of a set of starting points? In general, it is important to try different sets of starting points when doing k-means.

ANSWER

From the previous two questions, if I were to choose one, I would choose the second set of starting points which are (-3,-3), (2,2), and (-7,-7) which represents a clear clustering (good meaning in my opinion) between positive integer value and negative integer. Even though the point (-7, -7) is isolated from others, this clustering gives a clear interpretation. For example, we might consider the positive value as the machine performance where the higher the better and the negative like (-3, -3) or (-2, -4) might indicate that this machine should be fixed or repaired. An extreme case like (7,7) can be thought of as the serious case that should be maintained immediately otherwise the machine will explode!

```
def in_cluster_dist(clustered_data):
    total_dist = 0
    for c in clustered_data:
        for d in clustered_data[c]:
            total_dist += euclidean_dist(d, c)
    return total_dist
```

Figure 5: The algorithm that use for calculating the tightness in cluster

For the method that comes to my mind when I want to evaluate the K-means' starting points, I would use the simple way which is the sum of squared distance (Euclidean distance) between each point in a cluster to its own centroid then sum these values for all clusters as shown in Figure 5. This metric indicates that the lower the value the better the set of starting points or tighter clusters. From the given two previous questions, I calculated such distance and got the following result for each questions:

- **T5**: 13.270
- **T6**: 21.972

Once I know this information it would be a great criteria for me to choose which one. Right now, I would like to choose the first one from T5 even though the T6's reasons above are all about application side but this metric indicates that T5 gives tighter clusters.

OT2. What would be the best K for this question? Describe your reasoning.

ANSWER

In my opinion, the way to choose the number of K is to look at the plot of data if the data is simple like this but if the data contains a lot of information. Then, I might find the number of K by starting from a small number to larger number with the set of random starting points and use the Elbow metric to choose the optimal one. For this question, I might choose K to be 4, in which each centroid represents each group of data points which I plot below in Figure 6 where this K setting gives the result of the sum of distance within each cluster to be **6.83** where **T5** and **T6** got **13.27** and **21.97** respectively. Also, the more precise results are shown in Figure 7.

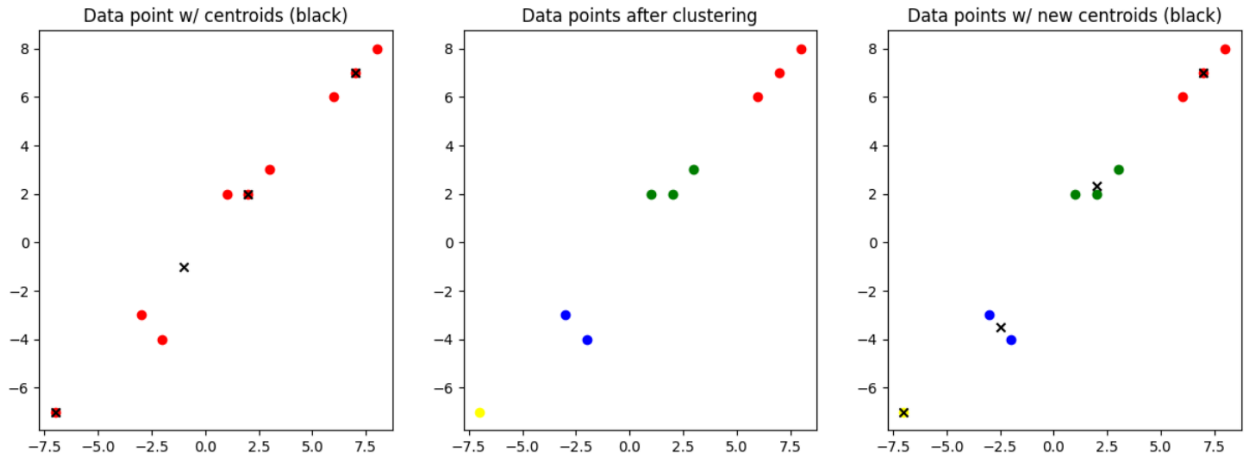


Figure 6: The plot (after switching starting points and adding one more cluster(K)) of original data points in red color with their defined centroids in black color (left). After clustering, the K-means algorithm clearly distinguishes the data points into three groups (middle). The rightmost plot shows the updated centroids in black color. The set of starting points is (7,7), (2,2), (-1,-1) and (-7,-7).

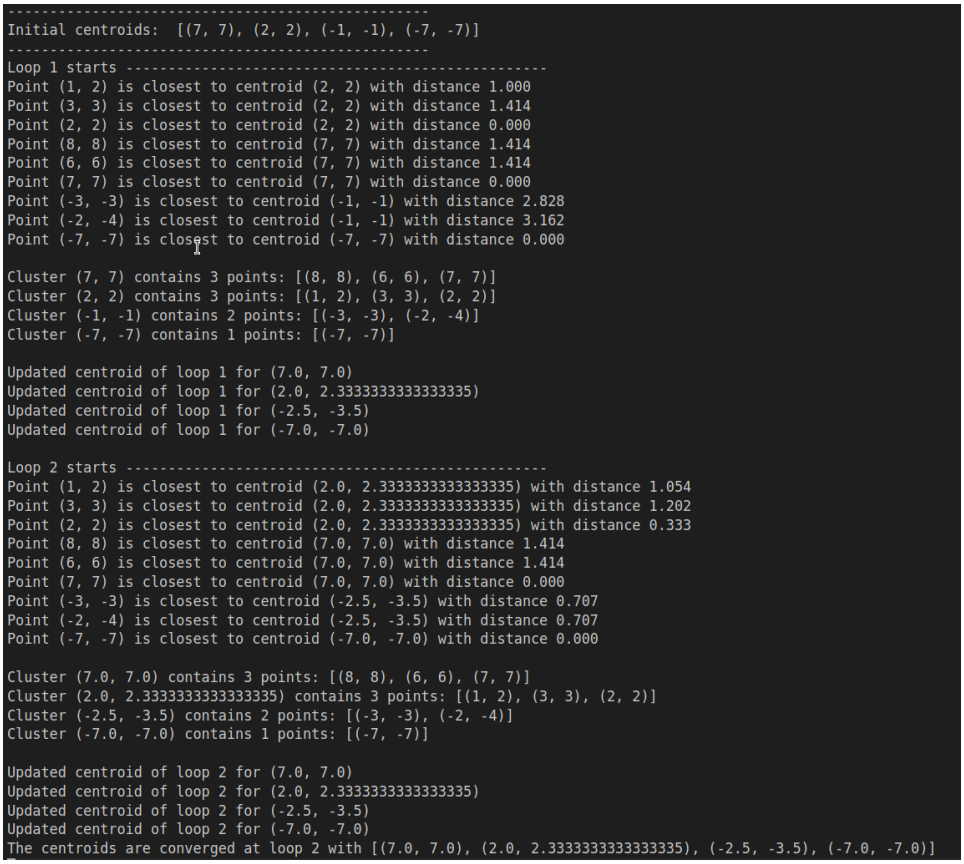


Figure 7: The result in the terminal shows how each data point is assigned and what the new centroids (mean) of each cluster after assigning all the points. Also, there is the final result of how each data point is in each cluster with its new centroids and the distance metric in **T7**.

T8. What is the median age of the training set?
ANSWER
The median **Age** of this training set is **28** when before filling the null value with mode. After filling all the null values with mode, the median of Age becomes **24**.

T9. What is the mode of Embarked?
ANSWER
The mode of **Embarked** after categorizing each feature from S to 0, C to 1, and Q to 2 is **0** which represents S. For **Sex**, they contain two categorical things which are "male" and "female" so I categorize each feature from "male" to 1 and "female" to 0. The mode for Sex is **1** which represents "male".

T10. Write a logistic regression classifier using gradient descent as learned in class. Use PClass, Sex, Age, and Embarked as input features.
ANSWER

Before implementing the model, I create the function that is used for cleaning, preparing, and normalizing the data as shown below and then load the data with four features and one label in Figure 8 which the full code will be uploaded in separate files.

```
> def clean_data(df, columns=["Age", "Embarked", "Sex",]): …
> def _normalize_minmax(data, eps=1e-8): …
> def normalize(x, min=None, max=None, eps=1e-8): …
> def setup_train_data(xy, attr_to_normalize): …
> def inference(test, classifier): …
```

```
# Gather trainset for each experiment

# Common trainset for T10 and T12
trainset_00 = np.array(train_df[["Pclass","Sex","Age","Embarked", "Survived"]].values, dtype=float)

# T10 (Original)
trainset_01 = deepcopy(trainset_00)
trainset_01, trainset_01_stats = setup_train_data(trainset_01, [2,]) # Normalize only age
```

Figure 8: The code shows the functions used for preparing, cleaning and normalizing the data and loading the dataset for specific features.

After that, I implemented the Logistic Regression classifier which contains the following methods as shown in Figure 9.

```python
# Implement the logistic regression classifier
class LogisticRegClassifier:
    def __init__(self, learning_rate=0.01, epoch=1000, n_attributes=4, batch_size=32, val_perc=0.1):

    def setup_train_data(self, xy):


    def initialize(self,):

    def sigmoid(self, x):

    def forward(self, x, is_train=True):

    def inference(self, x):


    def calculate_loss(self, y, y_pred, eps=1e-8):

    def backward(self, x, y, y_pred):


    def calculuate_accuracy(self, y, y_pred):

    def train_without_val(self,):

    def train(self, val_perc=0.1):
```

Figure 9: The overall structure of the Logistic Regression Classifier class.

T11. Submit a screenshot of your submission (with the scores). Upload your code to courseville.
ANSWER

The screenshot for the above setting with 4 features which are Pclass, Sex, Age and Embarked has the following result in Figure 10 for train and test (in Kaggle):

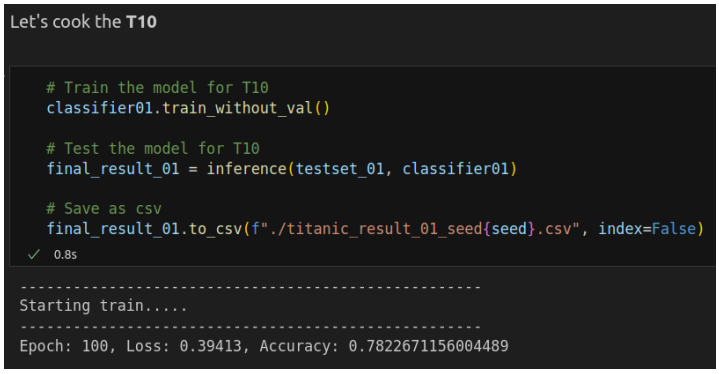


Figure 10: The evaluation score for train (above) and test set (below)

My setting for this experiment are as follow:

- Learning Rate: 0.01
- Epoch: 100
- Batch size: 64

T12. Try adding some higher order features to your training (x21 , x1 x2 ,...). Does this model has better accuracy on the training set? How does it perform on the test set?
ANSWER

I add the following features to the model which are added to the previous 4 which results in 7 in total:

- Pclass squared
- Age squared
- Sex multiply Age

The evaluation in the training set was quite good (better than only 4 features) but when evaluated on the test set. The model performed very similarly with just 4 features as shown in Figure 11.

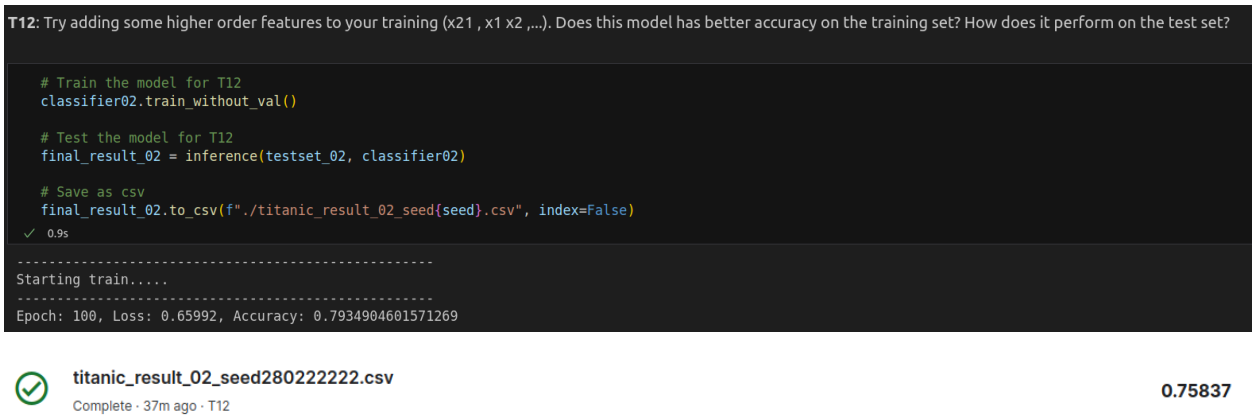


Figure 11: The training result (above) and the test result (below).

T13. What happens if you reduce the amount of features to just Sex and Age?
ANSWER

It is very surprising that the training set's accuracy and testing set's accuracy are better than and similar to the one that uses 4 features and even 7 features (better than this model!) with high order features by using only 2 features (Sex and Age) as shown in Figure 12.

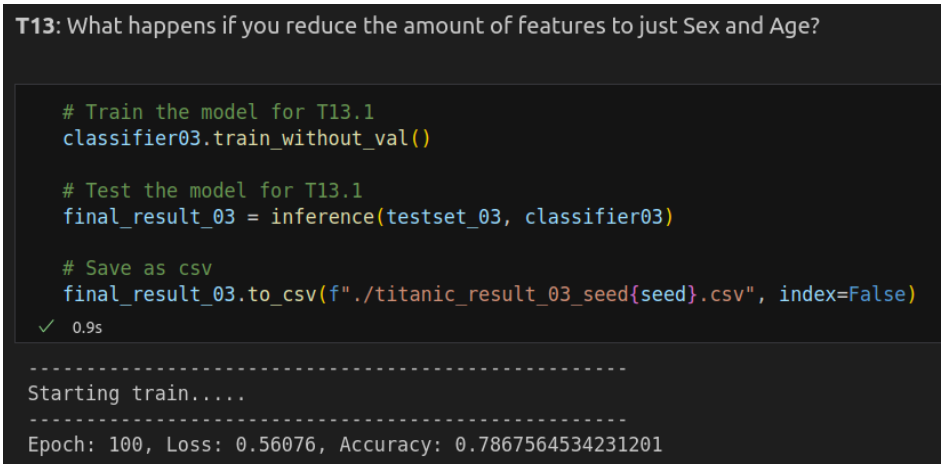


Figure 12: The results of a model that is trained using only 2 features (Age and Sex).

OT3. We want to show that matrix inversion yields the same answer as the gradient descent method. However, there is no closed form solution for logistic regression. Thus, we will use normal linear regression instead. Re-do the Titanic task as a regression problem by using linear regression. Use the gradient descent method.

ANSWER

I reimplemented the model by first removing the logistic function that is applied before outputting the result of the Logistic Regression model as shown in Figure 13.

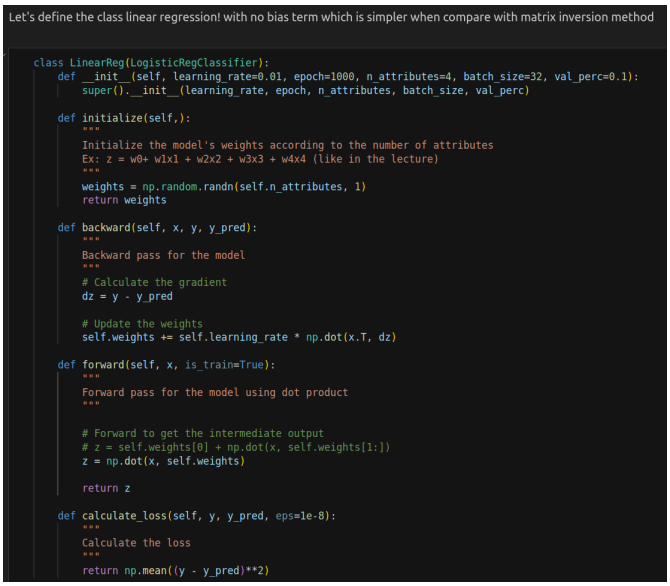


Figure 13: The implementation of Linear Regression with some modification from the Logistic model we used before.

In the following Figure 14, the results of two different settings for Linear Regression models which use different learning rates. The one that uses a higher learning rate has encountered the square's overflow but the other one doesn't. However, the model doesn't perform that well by using this model.

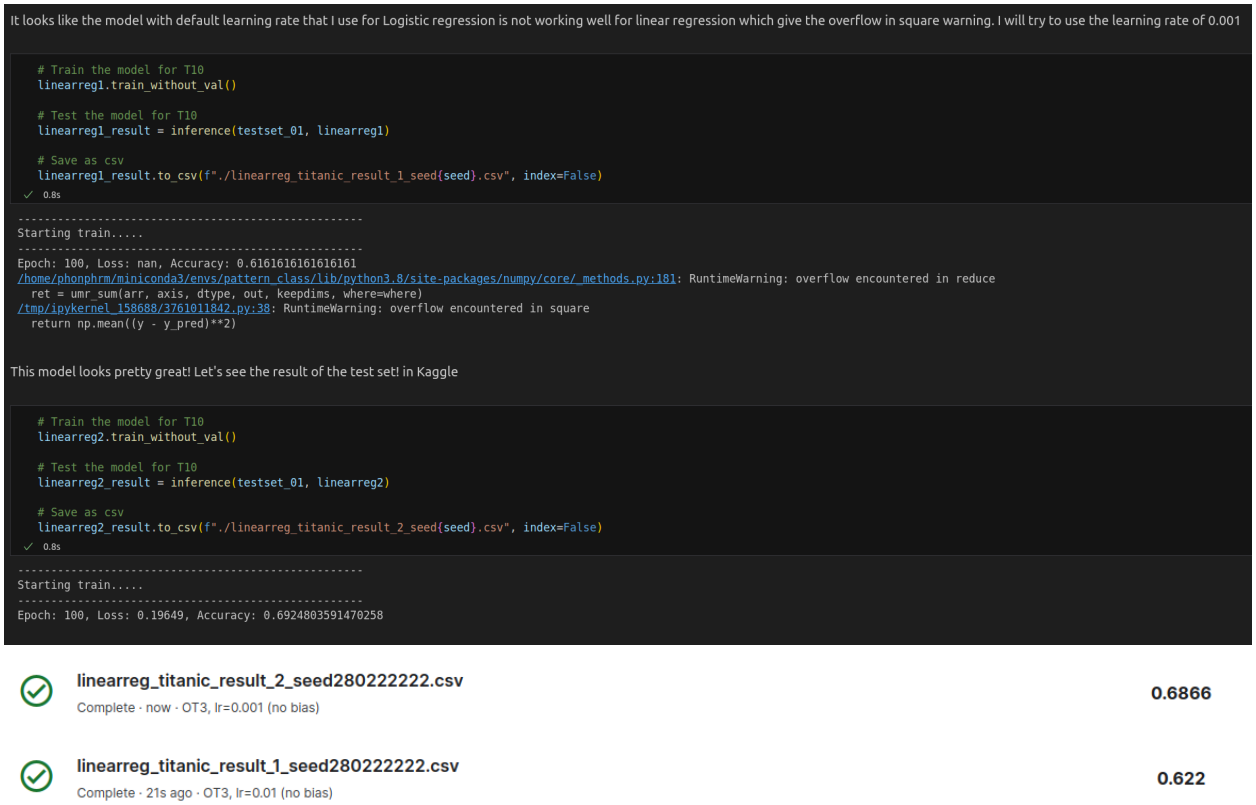


Figure 14: The result from training set for both learning rate setting and the testing result in Kaggle.

OT4. Now try using matrix inversion instead. However Are the weights learned from the two methods similar? Report the Mean Squared Errors (MSE) of the difference between the two weights.

ANSWER

The implementation of this method and comparing the difference are shown in Figure 15. The results show that the weight between two methods aren't that much different (quite the same!).

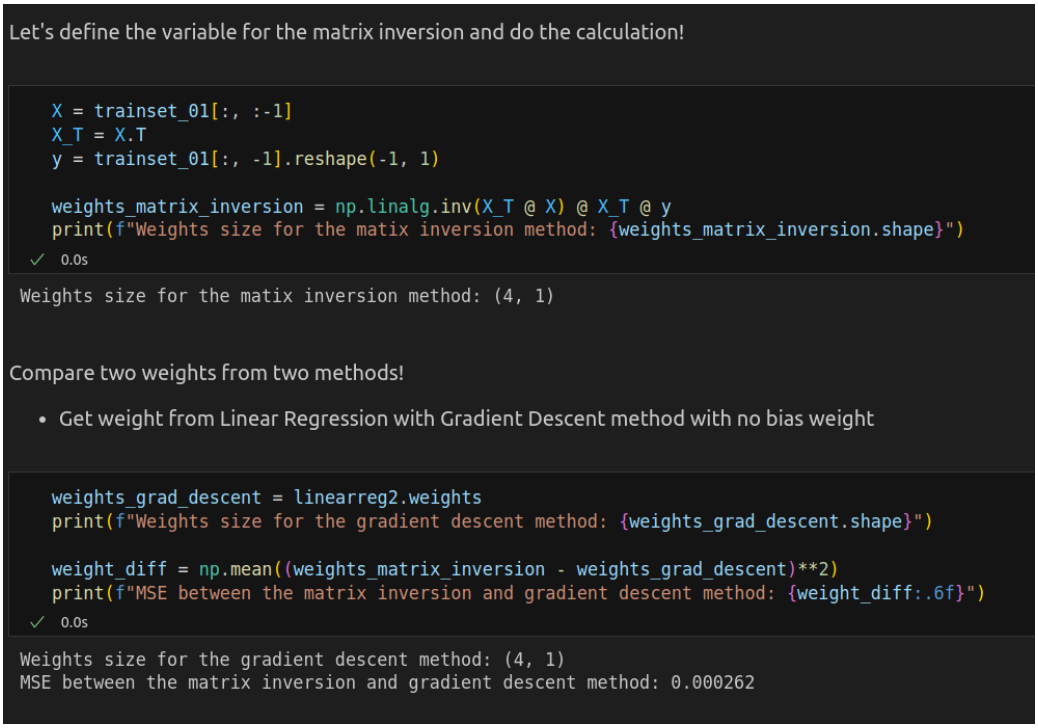


Figure 15: The implementation and comparison of matrix inversion and linear regression.