HW1 Clustering and Regression

6332032921 Pisitpong Chongpipattanakul

January 23, 2024

Contents

1 Metric

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

1.1 T1

$$\begin{aligned} Model\ A\ Accuracy &= \frac{Correct_{dog} + Correct_{cat}}{All} \\ &= \frac{30+40}{30+20+10+40} \\ &= \frac{70}{100} \\ &= 0.7 \end{aligned} \tag{1}$$

1.2 T2

In this problem we will consider cat as class 1 (positive).

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ &= \frac{40}{60} \\ &= \frac{2}{3} \end{aligned} \tag{2}$$

$$\begin{aligned} Recall &= \frac{TP}{TP+FN} \\ &= \frac{40}{50} \\ &= 0.8 \end{aligned} \tag{3}$$

$$\begin{aligned} F1\ Score &= 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \\ &= 2 \cdot \frac{\frac{2}{3} \cdot 0.8}{\frac{2}{3} + 0.8} \\ &= 2 \cdot \frac{1.6}{2+2.4} \\ &= 2 \cdot \frac{1.6}{4.4} \\ &= \frac{8}{11} \end{aligned} \tag{4}$$

1.3 T3

In this problem we will consider dog as class 1 (positive).

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ &= \frac{30}{40} \\ &= 0.75 \end{aligned} \tag{5}$$

$$
\begin{aligned}
Recall &= \frac{TP}{TP+FN} \\
&= \frac{30}{50} \\
&= 0.6
\end{aligned} \tag{6}
$$

$$
\begin{aligned}
F1\ Score &= 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \\
&= 2 \cdot \frac{0.75 \cdot 0.6}{0.75 + 0.6} \\
&= 2 \cdot \frac{0.45}{1.35} \\
&= \frac{2}{3}
\end{aligned} \tag{7}
$$

1.4 T4

Let's assume that the prediction is scaling in linear trend so value in actual dog will be multiplied by 0.4 and value in actual cat will be multiplied by 1.6

The result classification table after scaling

Model B	Predicted dog	Predicted cat
Actual dog	12	8
Actual cat	16	64

$$
\begin{aligned}
Model\ B\ Accuracy &= \frac{Correct_{dog} + Correct_{cat}}{All} \\
&= \frac{12+64}{12+8+16+64} \\
&= \frac{78}{100} \\
&= 0.78
\end{aligned} \tag{8}
$$

And we assign dog as the positive class

$$
\begin{aligned}
Precision &= \frac{TP}{TP+FP} \\
&= \frac{12}{12+16} \\
&= \frac{3}{7}
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
Recall &= \frac{TP}{TP+FN} \\
&= \frac{12}{20} \\
&= 0.6
\end{aligned} \tag{10}
$$

$$
\begin{aligned}
F1\ Score &= 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \\
&= 2 \cdot \frac{\frac{3}{7} \cdot 0.6}{\frac{3}{7} + 0.6} \\
&= 2 \cdot \frac{1.8}{7.2} \\
&= 0.5
\end{aligned} \tag{11}
$$

We see that only recall value remains unchanged but F1 score and precision are changed because ratio between class 1 and class 0 are changed so the value of F1 score and precision which also reference to class 0 also have affected

1.5 OT1

First, we will reform the equation of Accuracy and F1 Score

$$\begin{aligned} Accuracy &= \frac{TP+TN}{TP+TN+FP+FN} \\ &= \frac{1}{1+\frac{FP+FN}{TP+TN}} \end{aligned} \tag{12}$$

$$\begin{aligned} F1\ Score &= \frac{2TP}{2TP+FP+FN} \\ &= \frac{1}{1+\frac{FP+FN}{2TP}} \end{aligned} \tag{13}$$

The factor which affects differently between F1 Score and Accuracy are TP and TN so there are three cases

- If TP > TN then Accuracy < F1 Score
- If TP = TN then Accuracy = F1 Score
- If TP < TN then Accuracy > F1 Score

2 Hello Clustering

2.1 T5

```
# Code for T5, T6 and T7
import matplotlib.pyplot as plt
import numpy as np

COLOR = ['red', 'brown', 'orange']

# Assign Step
def distance(x, y, x_starting_point, y_starting_point):
    return np.sqrt((x-x_starting_point)**2 + (y-y_starting_point)**2)

def find_closest(x, y, x_starting_point, y_starting_point):
    distances = distance(x, y, x_starting_point, y_starting_point)
    return np.argmin(distances)

def assignToNewCentroid(x, y, x_centroid, y_centroid):
    assigned_list = np.array([], dtype=int)
    for i in range(len(x)):
        closest = find_closest(x[i], y[i], x_centroid, y_centroid)

        assigned_list = np.append(assigned_list, closest)

    return assigned_list

# Update Centroid
def calculateNewCentroid(x, y, assigned_list):
    x_starting_point = np.array([], dtype=float)
    y_starting_point = np.array([], dtype=float)

    for i in range(np.max(assigned_list) + 1):
        x_starting_point = np.append(x_starting_point, np.mean(x[assigned_list
            == i]))
        y_starting_point = np.append(y_starting_point, np.mean(y[assigned_list
            == i]))
```

```python
    return x_starting_point, y_starting_point

# Main kmeans calculation
def kmeans(xpoints, ypoints, x_centroid, y_centroid):
    assigned_list_prev = []

    while True:
        assigned_list = assignToNewCentroid(xpoints, ypoints, x_centroid,
            y_centroid)

        if len(assigned_list_prev) != 0:
            if np.array_equal(assigned_list, assigned_list_prev):
                break

        assigned_list_prev = assigned_list[:]

        for i in range(len(xpoints)):
            plt.scatter(xpoints[i], ypoints[i], color=COLOR[assigned_list[i]])
            print(f'Assign ({xpoints[i]}, {ypoints[i]}) to Centroid:
                ({x_centroid[assigned_list[i]]},
                {y_centroid[assigned_list[i]]})')

        x_centroid, y_centroid = calculateNewCentroid(xpoints, ypoints,
            assigned_list)

        for i in range(len(x_centroid)):
            plt.scatter(x_centroid[i], y_centroid[i], color='black')
            plt.annotate(f'New Centroid', (x_centroid[i], y_centroid[i]))
            print(f'New Centroid: ({x_centroid[i]}, {y_centroid[i]})')

        plt.show()

    return x_centroid, y_centroid, assigned_list
```

The first iteration has assigned the following points to each centroids

- Assign (1, 2) to Centroid: (2, 2)
- Assign (3, 3) to Centroid: (3, 3)
- Assign (2, 2) to Centroid: (2, 2)
- Assign (8, 8) to Centroid: (3, 3)
- Assign (6, 6) to Centroid: (3, 3)
- Assign (7, 7) to Centroid: (3, 3)
- Assign (-3, -3) to Centroid: (-3, -3)
- Assign (-2, -4) to Centroid: (-3, -3)
- Assign (-7, -7) to Centroid: (-3, -3)

And we get new three centroids as following point (purple dot)

- New Centroid: (6.0, 6.0)
- New Centroid: (1.5, 2.0)
- New Centroid: (-4.0, -4.666666666666667)

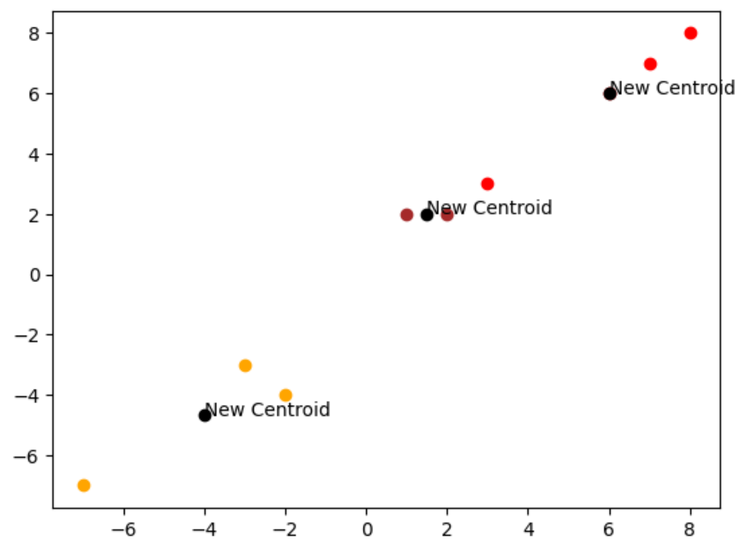


Figure 1: Graph after first k-means clustering iteration

The second iteration which also the last iteration has assigned the following points to each centroids

- Assign (1, 2) to Centroid: (1.5, 2.0)
- Assign (3, 3) to Centroid: (1.5, 2.0)
- Assign (2, 2) to Centroid: (1.5, 2.0)
- Assign (8, 8) to Centroid: (6.0, 6.0)
- Assign (6, 6) to Centroid: (6.0, 6.0)
- Assign (7, 7) to Centroid: (6.0, 6.0)
- Assign (-3, -3) to Centroid: (-4.0, -4.666666666666667)
- Assign (-2, -4) to Centroid: (-4.0, -4.666666666666667)
- Assign (-7, -7) to Centroid: (-4.0, -4.666666666666667)

And we get new three centroids as following point (purple dot)

- New Centroid: (7.0, 7.0)
- New Centroid: (2.0, 2.3333333333333335)
- New Centroid: (-4.0, -4.666666666666667)

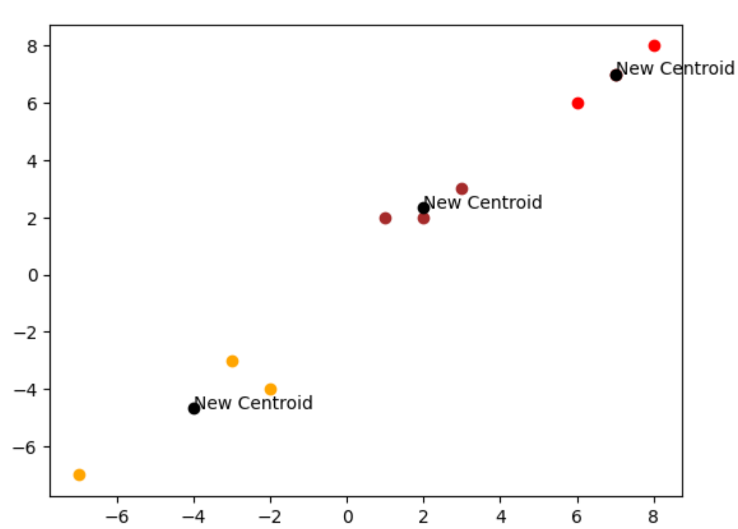


Figure 2: Graph after second k-means clustering iteration

2.2 T6

After changing the starting points to (-3, -3), (2, 2), (-7, -7) the following points are assigned to starting point

- Assign (1, 2) to Centroid: (2, 2)
- Assign (3, 3) to Centroid: (2, 2)
- Assign (2, 2) to Centroid: (2, 2)
- Assign (8, 8) to Centroid: (2, 2)
- Assign (6, 6) to Centroid: (2, 2)
- Assign (7, 7) to Centroid: (2, 2)
- Assign (-3, -3) to Centroid: (-3, -3)
- Assign (-2, -4) to Centroid: (-3, -3)
- Assign (-7, -7) to Centroid: (-7, -7)

And we get new three centroids as following point (purple dot)

- New Centroid: (-2.5, -3.5)
- New Centroid: (4.5, 4.666666666666667)
- New Centroid: (-7.0, -7.0)

we also see that centroid points have converged to stable point faster than T5 question and cluster is more grouping compared to T5 question

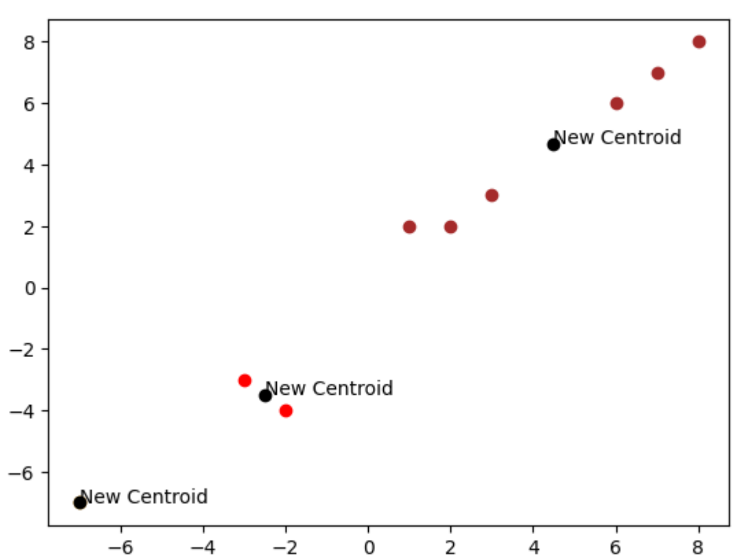


Figure 3: Graph after running k-means clustering

2.3 T7

I think the starting point from T5 is better than T6. I have measured "goodness" by using arithmetic mean of the euclidean distance between centroid and point in each cluster If we consider the arithmetic mean for T5 and T6 we will get that

- T5: Arithmetic mean of distance = 1.4744582139275695
- T6: Arithmetic mean of distance = 2.4413195239469854

which means the starting point from T6 question causes more sparse result cluster compared to starting point from T5 question

3 My heart will go on

3.1 Jupyter Notebook (T8-T13,OT3,OT4)

In [ ]:
```
%pip install matplotlib numpy pandas
```

```
Requirement already satisfied: matplotlib in e:\practice\pattern_2024\venv\lib\site-packag
es (3.8.2)
Requirement already satisfied: numpy in e:\practice\pattern_2024\venv\lib\site-packages
(1.26.3)
Requirement already satisfied: pandas in e:\practice\pattern_2024\venv\lib\site-packages
(2.1.4)
Requirement already satisfied: kiwisolver>=1.3.1 in e:\practice\pattern_2024\venv\lib\site
-packages (from matplotlib) (1.4.5)
Requirement already satisfied: python-dateutil>=2.7 in e:\practice\pattern_2024\venv\lib\s
ite-packages (from matplotlib) (2.8.2)
Requirement already satisfied: pyparsing>=2.3.1 in e:\practice\pattern_2024\venv\lib\site-
packages (from matplotlib) (3.1.1)
Requirement already satisfied: fonttools>=4.22.0 in e:\practice\pattern_2024\venv\lib\site
-packages (from matplotlib) (4.47.2)
Requirement already satisfied: packaging>=20.0 in e:\practice\pattern_2024\venv\lib\site-p
ackages (from matplotlib) (23.2)
Requirement already satisfied: cycler>=0.10 in e:\practice\pattern_2024\venv\lib\site-pack
ages (from matplotlib) (0.12.1)
Requirement already satisfied: pillow>=8 in e:\practice\pattern_2024\venv\lib\site-package
s (from matplotlib) (10.2.0)
Requirement already satisfied: contourpy>=1.0.1 in e:\practice\pattern_2024\venv\lib\site-
packages (from matplotlib) (1.2.0)
Requirement already satisfied: pytz>=2020.1 in e:\practice\pattern_2024\venv\lib\site-pack
ages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in e:\practice\pattern_2024\venv\lib\site-pa
ckages (from pandas) (2023.4)
Requirement already satisfied: six>=1.5 in e:\practice\pattern_2024\venv\lib\site-packages
(from python-dateutil>=2.7->matplotlib) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 23.0.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
train_url = "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv"
train = pd.read_csv(train_url)  # training set

test_url = "http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv"
test = pd.read_csv(test_url)  # test set
```

In [ ]:
```
print(train.head())
print(train.tail())
```

```
   PassengerId  Survived  Pclass  \
0            1         0       3   
1            2         1       1   
2            3         1       3   
3            4         1       1   
4            5         0       3   

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1   
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1   
2                             Heikkinen, Miss. Laina  female  26.0      0   
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1   
4                           Allen, Mr. William Henry    male  35.0      0   

   Parch            Ticket     Fare Cabin Embarked  
0      0         A/5 21171   7.2500   NaN        S  
1      0          PC 17599  71.2833   C85        C  
2      0  STON/O2. 3101282   7.9250   NaN        S  
3      0            113803  53.1000  C123        S  
4      0            373450   8.0500   NaN        S  
     PassengerId  Survived  Pclass                                      Name  \
886          887         0       2                     Montvila, Rev. Juozas   
887          888         1       1              Graham, Miss. Margaret Edith   
888          889         0       3  Johnston, Miss. Catherine Helen "Carrie"   
889          890         1       1                     Behr, Mr. Karl Howell   
890          891         0       3                       Dooley, Mr. Patrick   

        Sex   Age  SibSp  Parch      Ticket   Fare Cabin Embarked  
886    male  27.0      0      0      211536  13.00   NaN        S  
887  female  19.0      0      0      112053  30.00   B42        S  
888  female   NaN      1      2  W./C. 6607  23.45   NaN        S  
889    male  26.0      0      0      111369  30.00  C148        C  
890    male  32.0      0      0      370376   7.75   NaN        Q  
```

In [ ]:
```python
train.describe()
```

Out[ ]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

T8

In [ ]:
```python
# T8
median_age = train["Age"].median()

print(f"Median of age is {median_age}")

train["Age"] = train["Age"].fillna(train["Age"].median())
```

```
Median of age is 28.0
```

T9

In [ ]:

```
# T9
train["Embarked"] = train["Embarked"].fillna(train["Embarked"].mode()[0])

train.loc[train["Embarked"] == "S", "Embarked"] = 0
train.loc[train["Embarked"] == "C", "Embarked"] = 1
train.loc[train["Embarked"] == "Q", "Embarked"] = 2

train.loc[train["Sex"] == "male", "Sex"] = 0
train.loc[train["Sex"] == "female", "Sex"] = 1
```

T10

In [ ]:

```
features = np.array(train[["Pclass", "Sex", "Age", "Embarked"]].values, dtype=float)
results = np.array(train["Survived"].values, dtype=float)
features, results
```

```
Out[ ]: (array([[ 3.,  0., 22.,  0.],
        [ 1.,  1., 38.,  1.],
        [ 3.,  1., 26.,  0.],
        ...,
        [ 3.,  1., 28.,  0.],
        [ 1.,  0., 26.,  1.],
        [ 3.,  0., 32.,  2.]]),
 array([0., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 1., 0.,
        1., 0., 1., 0., 1., 1., 1., 0., 1., 0., 0., 1., 0., 0., 1., 1., 0.,
        0., 0., 1., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 1., 0., 0., 0.,
        0., 1., 1., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0., 0., 1., 1., 0.,
        1., 0., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 1., 1., 0., 1.,
        1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0.,
        0., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0., 1., 0., 0.,
        1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 1., 0., 0., 0., 0., 1., 0.,
        0., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 0., 0.,
        0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0., 1.,
        1., 0., 0., 1., 0., 1., 1., 1., 1., 0., 0., 1., 0., 0., 0., 0., 0.,
        1., 0., 0., 1., 1., 1., 0., 1., 0., 0., 0., 1., 1., 0., 1., 0., 1.,
        0., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 0., 0., 1.,
        0., 0., 0., 1., 0., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0.,
        1., 1., 1., 1., 1., 0., 1., 0., 0., 0., 0., 0., 1., 1., 1., 0., 1.,
        1., 0., 1., 1., 0., 0., 0., 1., 0., 0., 0., 1., 0., 0., 1., 0., 1.,
        1., 1., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 0., 1., 0., 1.,
        1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1., 0., 1., 1., 0., 0., 1.,
        1., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0.,
        1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1.,
        0., 1., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0.,
        0., 1., 1., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 0., 1., 1.,
        1., 0., 1., 1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 1.,
        0., 0., 0., 0., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0.,
        0., 1., 1., 0., 1., 1., 1., 1., 0., 0., 1., 0., 1., 0., 0., 1., 0.,
        0., 1., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 1., 1.,
        0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0.,
        0., 0., 0., 1., 0., 0., 0., 1., 1., 0., 1., 0., 0., 1., 0., 0., 0.,
        0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 1., 0., 1.,
        1., 0., 1., 1., 0., 0., 1., 0., 1., 0., 1., 0., 0., 1., 0., 0., 1.,
        0., 0., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1., 0., 0., 1.,
        0., 0., 1., 1., 0., 1., 1., 0., 0., 1., 1., 0., 1., 0., 1., 1., 0.,
        0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 0., 0., 1., 1.,
        0., 1., 1., 1., 0., 0., 0., 1., 0., 1., 0., 0., 0., 1., 0., 0., 0.,
        0., 1., 0., 0., 1., 1., 0., 0., 0., 1., 0., 0., 1., 1., 1., 0., 0.,
        1., 0., 0., 1., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0., 0., 1., 0.,
        0., 1., 0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 1., 0., 1., 1., 1.,
        0., 1., 0., 1., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 1., 0., 0.,
        0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 1., 0., 0., 0., 1., 0., 1.,
        0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 0., 0., 0., 0.,
        1., 0., 0., 1., 1., 0., 0., 0., 0., 1., 1., 1., 1., 1., 0., 1., 0.,
        0., 0., 1., 1., 0., 0., 1., 0., 0., 0., 1., 0., 1., 1., 0., 0., 1.,
        0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 0., 1., 0., 0., 1.,
        0., 0., 1., 1., 0., 0., 1., 1., 0., 0., 0., 1., 0., 0., 1., 1., 0.,
        1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 1., 0., 1., 1., 1.,
        0., 0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 1., 1., 0.,
        0., 0., 1., 1., 1., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.,
        0., 0., 0., 0., 1., 1., 0., 1., 0., 0., 0., 1., 1., 1., 1., 1., 0.,
        0., 0., 1., 0., 0., 1., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 1.,
        0., 0., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 0., 0., 1., 1.,
        0., 0., 1., 0., 1., 0., 0., 1., 1., 0., 0., 0., 1., 1., 0., 0., 0.,
        0., 0., 0., 1., 0., 1., 0.]))
```

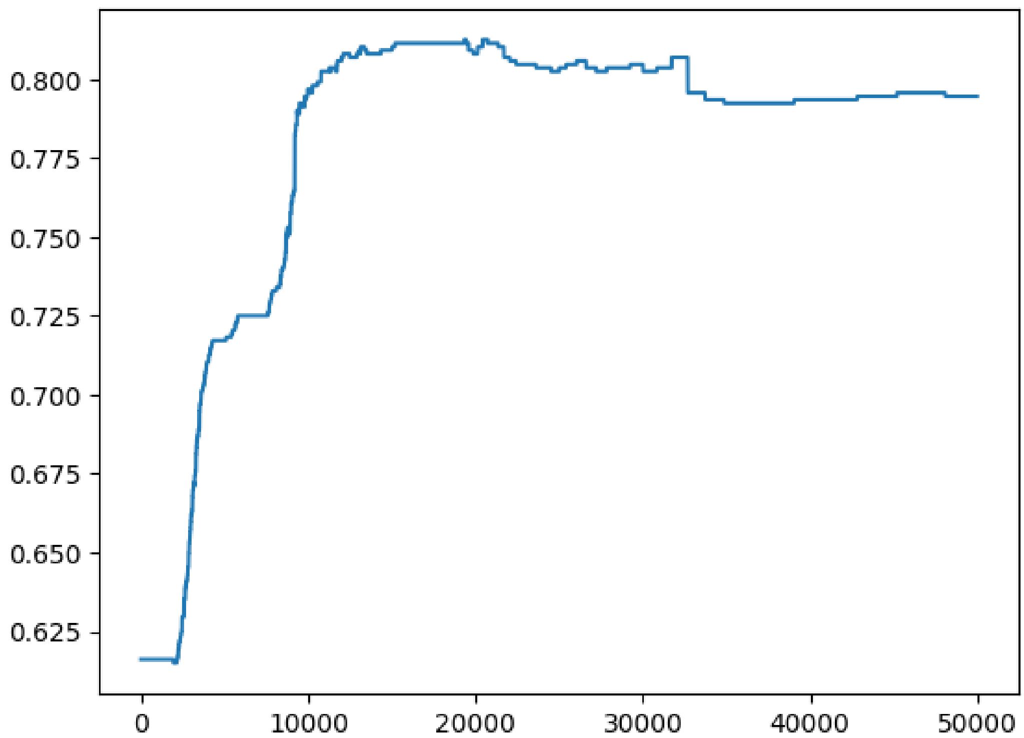
```
In [ ]: learning_rate = 1e-6


        def calculate_logistic(h):
            return 1 / (1 + np.exp(-h))
```

```python
def get_next_theta(theta, x, y):
    error = y - calculate_logistic(np.dot(x, theta))

    diff_theta = x.T.dot(error) * learning_rate

    return diff_theta


def classifier(theta, x):
    solution = calculate_logistic(np.dot(x, theta))

    classifier_result = solution >= 0.5

    solution[classifier_result == True] = 1
    solution[classifier_result == False] = 0

    return solution


def measurement(predicts, actual):
    diff = predicts - actual
    correct = diff == 0

    return np.sum(correct) / actual.shape


def train_logistic_regression(features, results):
    iterations = int(1e4)
    features_with_one = np.insert(features, 0, 1, axis=1)

    starting_theta = np.zeros(features_with_one.shape[1])
    theta = np.copy(starting_theta)
    accuracy_list = []

    for _ in range(iterations):
        theta += get_next_theta(theta, features_with_one, results)
        accuracy = measurement(
            np.array(train["Survived"]), classifier(theta, features_with_one)
        )[0]
        accuracy_list.append(accuracy)

    print(theta)
    print("Training Accuracy:", accuracy_list[-1])

    plt.plot(accuracy_list)
    plt.show()

    return theta
```

In [ ]:

```python
theta = train_logistic_regression(features, results)
```

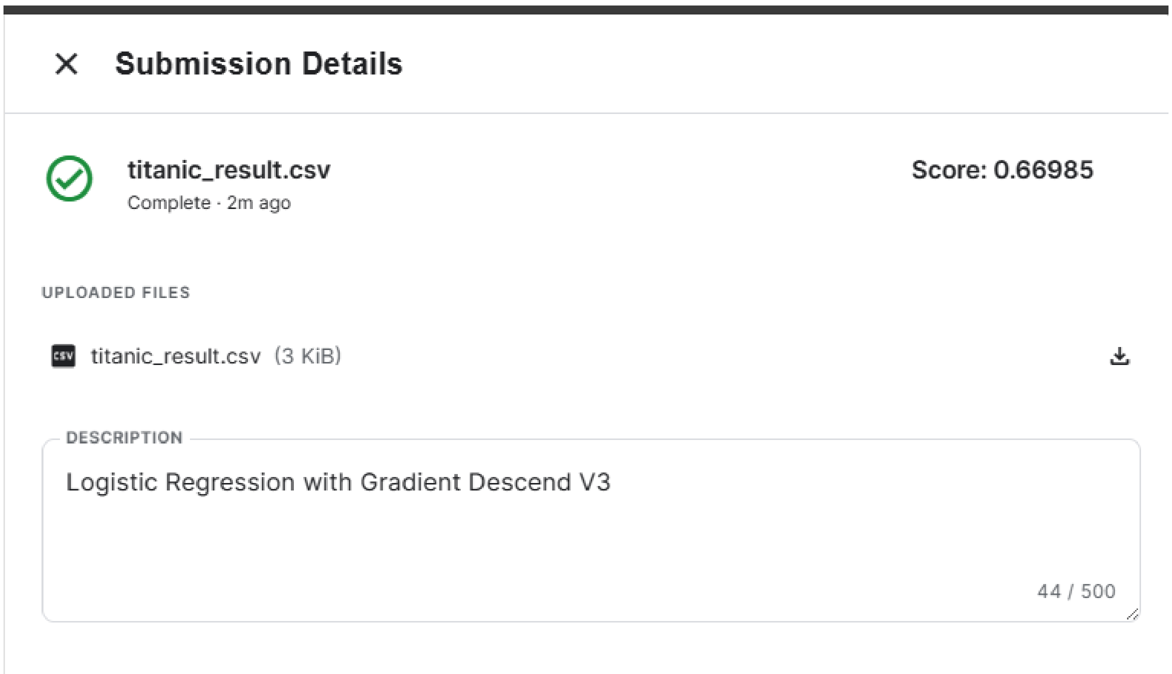
```
[ 0.60877697 -0.77117063  2.1963073  -0.01308992  0.34021432]
Training Accuracy: 0.7946127946127947
```



In [ ]:
```
# Clean test data
test["Embarked"] = test["Embarked"].fillna(test["Embarked"].mode()[0])

test.loc[test["Embarked"] == "S", "Embarked"] = 0
test.loc[test["Embarked"] == "C", "Embarked"] = 1
test.loc[test["Embarked"] == "Q", "Embarked"] = 2

test.loc[test["Sex"] == "male", "Sex"] = 0
test.loc[test["Sex"] == "female", "Sex"] = 1
```

T11

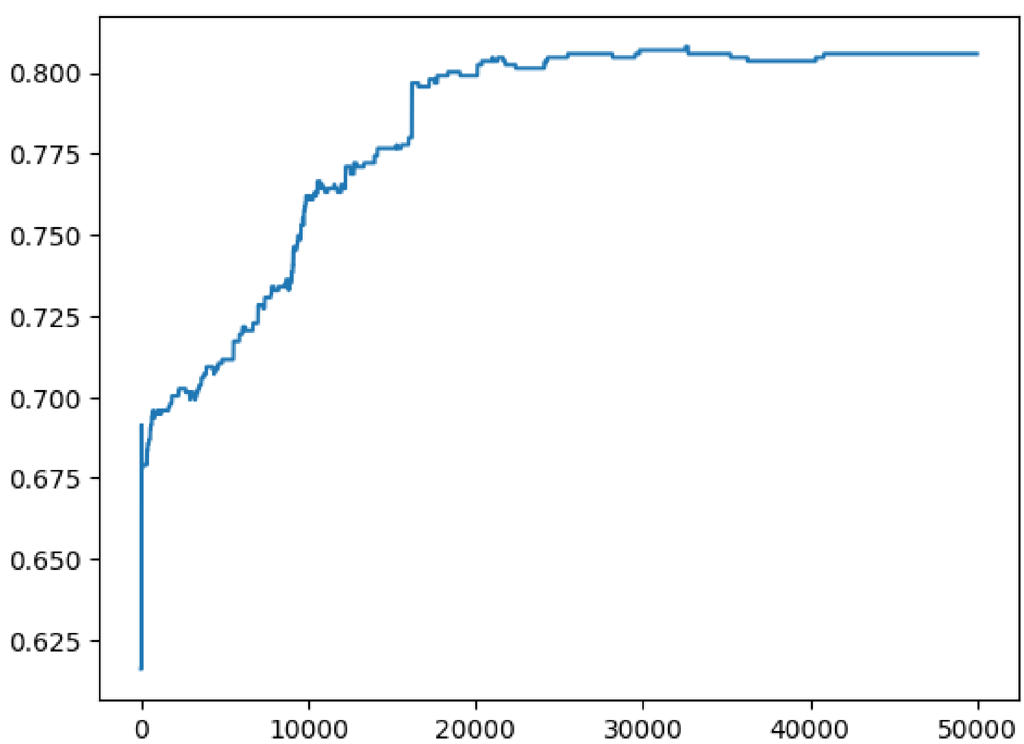
In [ ]:
```
# T11
test_features = np.array(test[["Pclass", "Sex", "Age", "Embarked"]].values, dtype=float)
test_features = np.insert(test_features, 0, 1, axis=1)

test_result = classifier(theta, test_features)
test_id = test[["PassengerId"]]

df = pd.DataFrame()

df["PassengerId"] = test_id
df["Survived"] = np.array(test_result, dtype=int)

df.to_csv("titanic_result.csv", index=False)
```



T12

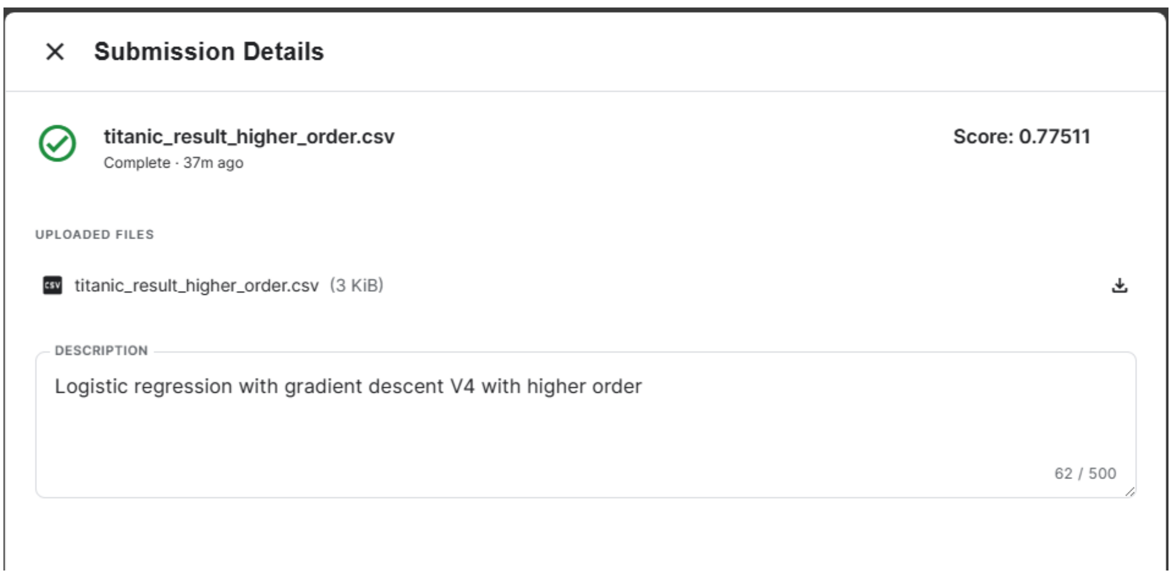
```
In [ ]: features_with_higher_order = np.insert(features, 0, features[:, 0] ** 2, axis=1)
features_with_higher_order = np.insert(
    features_with_higher_order, 0, features[:, 0] * features[:, 2], axis=1
)

theta_higher_order = train_logistic_regression(features_with_higher_order, results)
```

```
[ 0.27778632 -0.01063309 -0.23897882  0.22627652  2.13072319 -0.00735009
  0.36545214]
Training Accuracy: 0.8058361391694725
```




```
In [ ]: test_features = np.array(test[["Pclass", "Sex", "Age", "Embarked"]].values, dtype=float)

test_features_higher_order = np.insert(
    test_features, 0, test_features[:, 0] ** 2, axis=1
)
```

```python
test_features_higher_order = np.insert(
    test_features_higher_order, 0, test_features[:, 0] * test_features[:, 2], axis=1
)
test_features_higher_order = np.insert(test_features_higher_order, 0, 1, axis=1)

test_result = classifier(theta_higher_order, test_features_higher_order)
test_id = test[["PassengerId"]]

df = pd.DataFrame()

df["PassengerId"] = test_id
df["Survived"] = np.array(test_result, dtype=int)

df.to_csv("titanic_result_higher_order.csv", index=False)
```

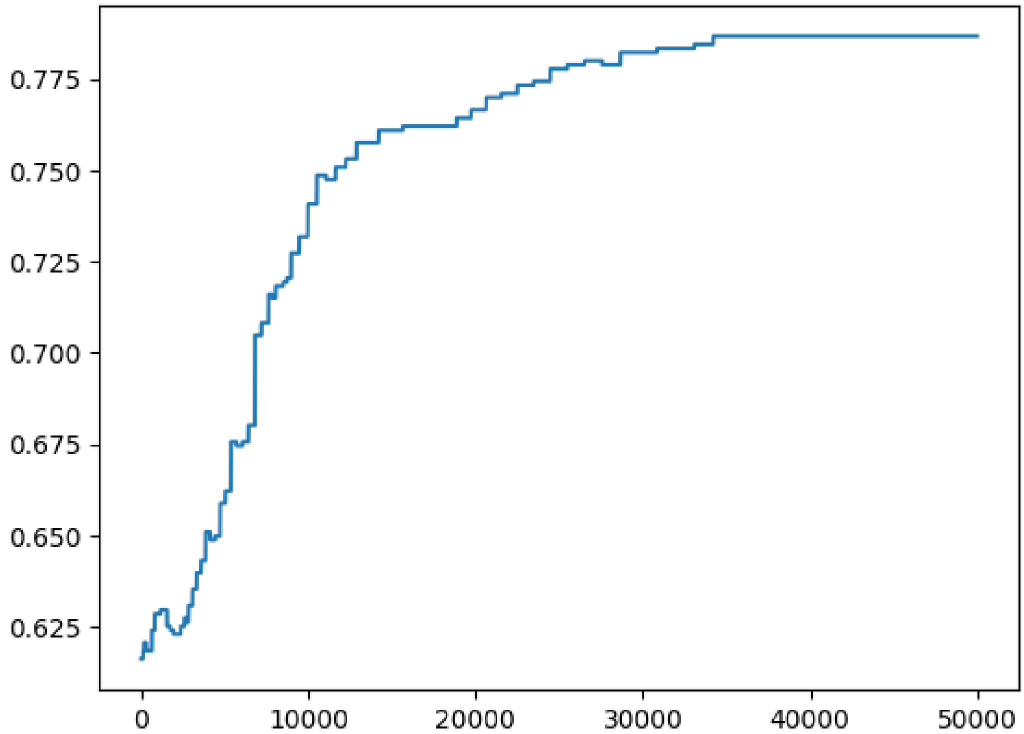


We see that there are little accuracy difference which higher order feature gives more accuracy than normal feature

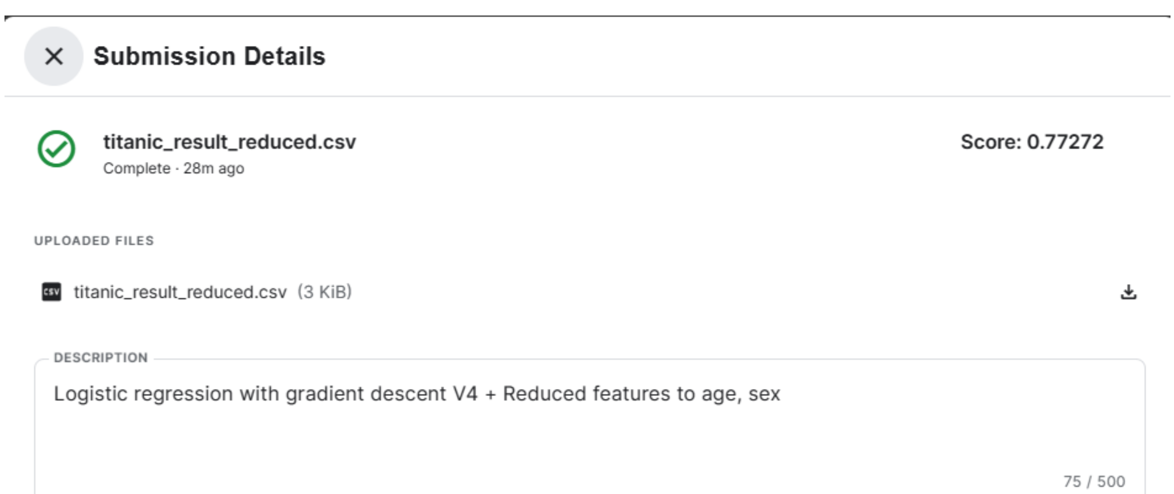
T13

In [ ]:
```python
reduced_features = np.array(train[["Sex", "Age"]].values, dtype=float)

theta_reduced = train_logistic_regression(reduced_features, results)
```

```
[-0.68387827  2.04999267 -0.0178297 ]
Training Accuracy: 0.7867564534231201
```



In [ ]:
```
test_features_reduced = np.array(test[["Sex", "Age"]].values, dtype=float)
test_features_reduced = np.insert(test_features_reduced, 0, 1, axis=1)

test_result = classifier(theta_reduced, test_features_reduced)
test_id = test[["PassengerId"]]

df = pd.DataFrame()

df["PassengerId"] = test_id
df["Survived"] = np.array(test_result, dtype=int)

df.to_csv("titanic_result_reduced.csv", index=False)
```



Overall result from reduced features give less accuracy than normal feature or features which has higher order

OT3

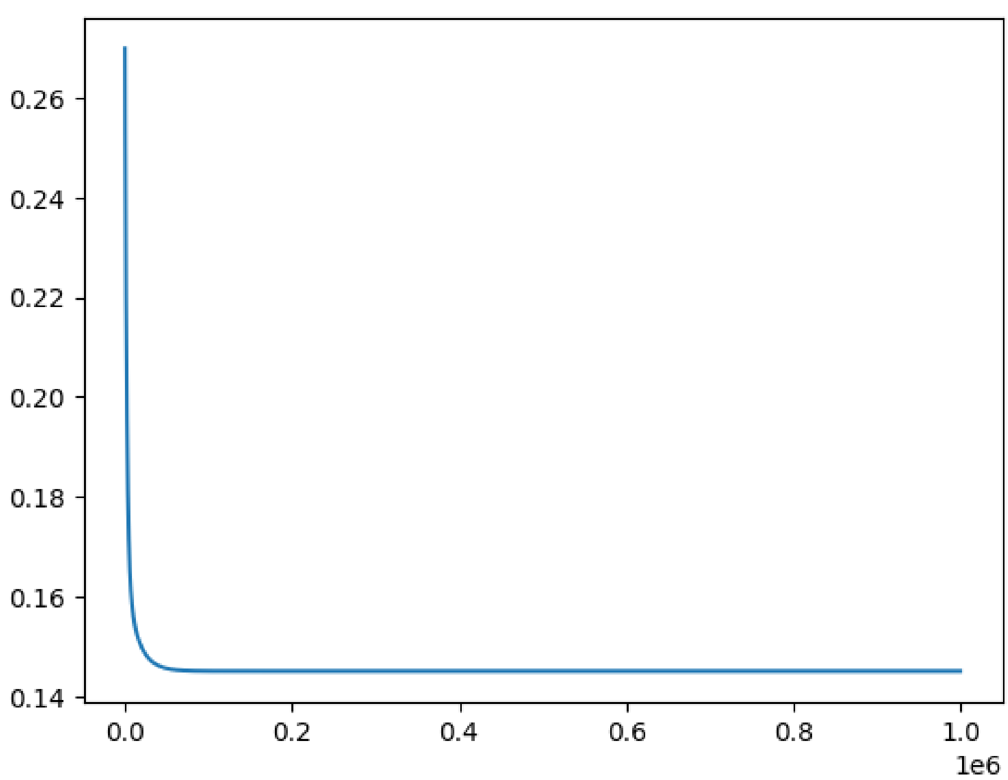
In [ ]:
```
# OT3
learning_rate = 1e-6
```

```python
def measurement(results, actual):
    size = results.shape[0]
    return np.sum((results - actual) ** 2) / size


def get_next_theta(theta, x, y):
    diff_theta = x.T.dot(y - x.dot(theta)) * learning_rate

    return diff_theta


def train_linear_regression(features, results):
    iterations = int(1e6)
    features_with_one = np.insert(features, 0, 1, axis=1)

    starting_theta = np.zeros(features_with_one.shape[1])
    theta = np.copy(starting_theta)

    accuracy_list = []

    for _ in range(iterations):
        diff_theta = get_next_theta(theta, features_with_one, results)
        theta += diff_theta
        accuracy = measurement(
            np.array(train["Survived"]), np.dot(features_with_one, theta)
        )
        accuracy_list.append(accuracy)

    plt.plot(accuracy_list)
    plt.show()

    print("Mean Square Error:", accuracy_list[-1])
    return theta
```

In [ ]:

```python
linear_theta = train_linear_regression(features, results)
linear_theta
```

```
Mean Square Error: 0.1449257376613481
```



Out[ ]:

```
array([ 0.77654442, -0.18843944,  0.49086711, -0.00505436,  0.04911346])
```

Result

- Mean Square Error: 0.1449257376613481
- Parameter: [0.77654442, -0.18843944, 0.49086711, -0.00505436, 0.04911346]

OT4

In [ ]:
```
# OT4

features_copy = np.insert(features, 0, 1, axis=1)

inverse_props = np.linalg.inv(np.matmul(features_copy.transpose(), features_copy))
linear_reg_theta = np.matmul(
    np.matmul(inverse_props, features_copy.transpose()), results
)

loss = measurement(np.array(train["Survived"]), np.dot(features_copy, linear_reg_theta))

print("Mean Square Error:", loss)
print("Parameter:", linear_reg_theta)
```

```
Mean Square Error: 0.14492573766134811
Parameter: [ 0.77654442 -0.18843944  0.49086711 -0.00505436  0.04911346]
```

Result

- Mean Square Error: 0.14492573766134811
- Parameter: [0.77654442 -0.18843944 0.49086711 -0.00505436 0.04911346]

which gives the same MSE and parameter value as in OT3

3.2 OT5

Show that $\nabla_A trAB = B^T$

Because tr can be used only if AB is a square matrix so assume that A is a matrix size $N * M$ and B is a matrix size $M * N$

$$\begin{aligned} X &= \nabla_A trAB \\ X_{i,j} &= \frac{\partial \sum_{a=1}^{a=N} \sum_{b=1}^{b=M} A_{a,b}B_{b,a}}{\partial A_{i,j}} \\ &= \frac{\partial A_{i,j}B_{j,i}}{\partial A_{i,j}} \\ &= B_{j,i} \\ \therefore \nabla_A trAB = X &= B^T \end{aligned} \tag{14}$$

3.3 OT6

Show that $\nabla_{A^T} f(A) = (\nabla_A f(A))^T$

Assume that A is a matrix size $N * M$ and let $X = \nabla_A f(A)$, $Y = \nabla_{A^T} f(A)$

$$\begin{aligned} Y_{i,j} &= \frac{\partial f(A)}{\partial A_{j,i}} \\ &= X_{j,i} \\ \therefore \nabla_{A^T} f(A) = Y &= X^T = (\nabla_A f(A))^T \end{aligned} \tag{15}$$