HW3 Answers

March 31, 2023

Completed by: Koh Aik Hong (1005139)

[ ]:
```
from IPython.display import display, Image
```

1 Logistic Regression [25 points]

Consider 13 data points from a 2-d space where each point is of the form $x = (x_1, x_2)$, as shown in Figure 1. Now we want to train a logistic regression classifier based on the given data. Suppose the hypothesis function of the logistic regression is $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$ where $g(z)$ is the logistic function, and parameter vector $\theta = [\theta_0, \theta_1, \theta_2]^T$ is initialized as $[0, -1, 1]^T$.

Please write a function implementing gradient descent (simultaneously updating all the parameters, i.e., $\theta_0, \theta_1, \theta_2$) to find suitable parameters for the hypothesis function. We set the learning rate as $= 0.1$ and run the gradient descent for 150 iterations. Plot the training loss over the 150 iterations using a line plot and write down the final decision boundary.

[ ]:
```
display(Image(filename="fig_1.png"))
```

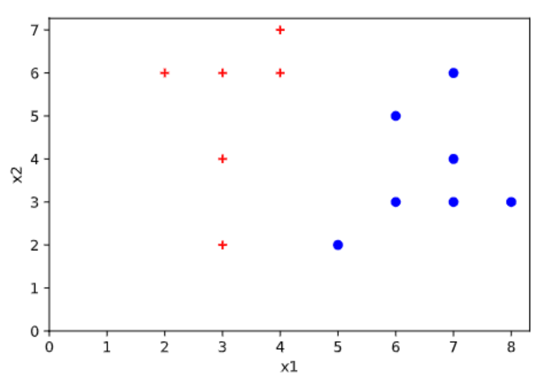


Figure 1: The given toy dataset contains 13 data points, where the red cross indicates that the point belongs to the **positive** class ($y = 1$), and the blue dot indicates that the point belongs to the **negative** class ($y = 0$).

[ ]:
```
dataset = [
  [4,7,1],
  [2,6,1],
  [3,6,1],
```

```
  [4,6,1],
  [3,4,1],
  [3,2,1],
  [7,6,0],
  [6,5,0],
  [7,4,0],
  [8,3,0],
  [7,3,0],
  [6,3,0],
  [5,2,0]
]
```

[ ]:
```
import numpy as np

# Convert dataset to numpy array
dataset = np.array(dataset)
X = dataset[:,:-1]
y = dataset[:,-1]

# insert a column of ones to X
X = np.append(np.ones((X.shape[0],1)),X, axis=1)
print("X = ")
print(X[0:3],"\n")
print("y = ")
print(y[0:3])
```

```
X =
[[1. 4. 7.]
 [1. 2. 6.]
 [1. 3. 6.]]

y =
[1 1 1]
```

[ ]:
```
# implement the hypothesis function for logistic regression
def hypothesis_function(x, w):
  z = x @ w
  return 1/(1+np.exp(-z))
```

[ ]:
```
# implement the cost function for logistic regression
def cost_function(x, y, w):
  n = x.shape[0]
  h = hypothesis_function(x,w)
  cost = -1/n * (y @ np.log(h) + (1 - y) @ np.log(1 - h))
  return cost
```

[ ]:
```python
def gradient_descent(X, y, w, alpha, iterations):
  # add in the initial loss
  n = X.shape[0]
  losses = np.zeros(iterations+1)
  # add in initial loss before starting gradient descent
  losses[0] = cost_function(X,y,w)
  for i in range(iterations):
    h = hypothesis_function(X,w)
    # using the gradient update rule in the slides without the 1/n according to␣
 ↪clarifications with prof Zhao Na
    w = w - alpha * (h - y)@ X
    loss = cost_function(X,y,w)
    losses[i+1] = loss

  return w, losses
```

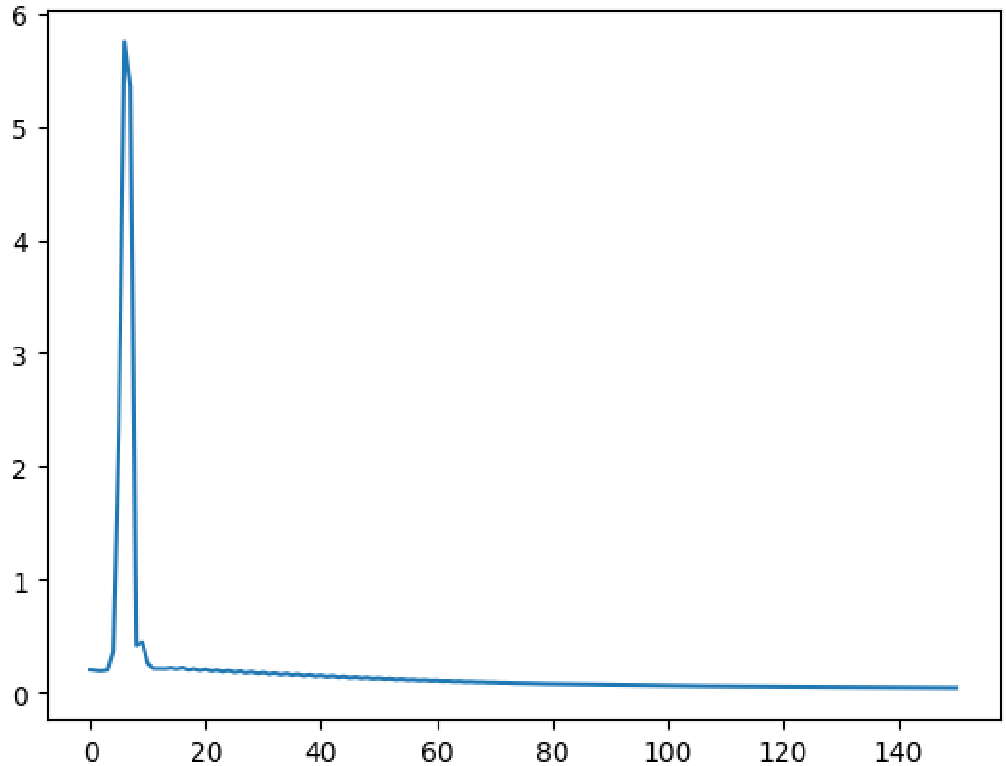
[ ]:
```python
# initialize the weights
initial_w = np.array([0,-1,1])
print("initial weights: ",initial_w)

# train the model
w, losses = gradient_descent(X, y, initial_w, 0.1, 150)
print("final weights: ",w)
```

```
initial weights:  [ 0 -1  1]
final weights:  [ 5.17186191 -2.41702311  1.44246911]
```

[ ]:
```python
# plot the iteration against losses
import matplotlib.pyplot as plt
plt.plot(losses)
```

[ ]: [<matplotlib.lines.Line2D at 0x1d9fa671d20>]



1.1 Final decision boundary:

$h_\theta(x) = g(5.17186191 - 2.41702311x_1 + 1.44246911x_2)$

2 Neural Networks [40 points]

As shown in Figure 2, we are given a 3-layer neural network. Instead of incorporating the bias term into the weight matrix Θ_i, we explicitly write the bias term, resulting in $z_2 = w_2 a_1 + b_2$, where $w_2 \in R^{1\times 3}$ is the weight vector and $b_2 \in R$ is the bias term for the mapping function from the hidden layer to the output layer, respectively. The output of this neural network $\hat{y} = \sigma(z_2)$.

(1) [5 points] Write down the mathematical representation of $\frac{\partial \hat{y}}{\partial w_2}$ and $\frac{\partial \hat{y}}{\partial b_2}$.

(2) [5 points] Let us assume the activation function   is the softplus function, please derive the closed-form expression for calculating $\frac{\partial \hat{y}}{\partial w_2}$ and $\frac{\partial \hat{y}}{\partial b_2}$.

(3) [15 points] If we still use softplus as the activation function but change the value of the bias b_2, does $\frac{\partial \hat{y}}{\partial x}$ (x is the input to this 3-layer neural network) change? Please prove your answer. [Hint: the change of bias can be expressed as Δb_2]

(4) [15 points] Now let us assume that the activation function   is the logistic function, please derive the closed-form expression for calculating $\frac{\partial \hat{y}}{\partial w_1}$.

```
[ ]: display(Image(filename="fig_2.png"))
```

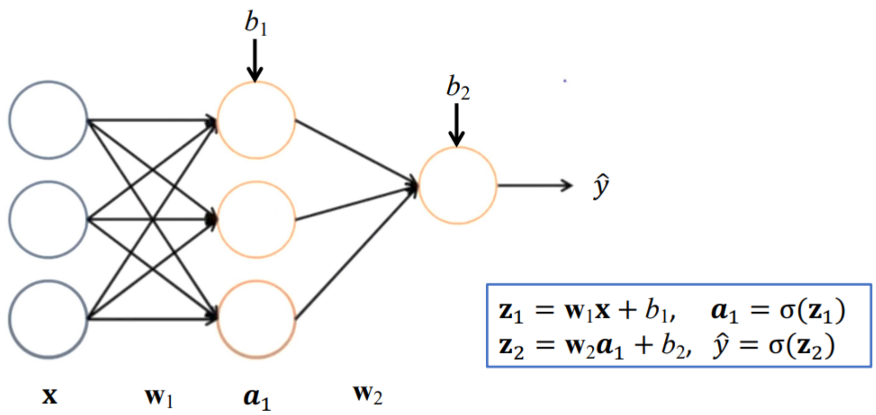


Figure 2: The architecture of a 3-layer neural network with the mathematical representation of forward propagation.

2.1 2(1) Answer

Given: $\hat{y} = \sigma(z_2) = \sigma(w_2 a_1 + b_2) = \sigma(w_2(\sigma(w_1 x + b_1) + b_2))$

$$\frac{\partial \hat{y}}{\partial w_2} = \frac{\partial \hat{y}}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$

$$\frac{\partial \hat{y}}{\partial b_2} = \frac{\partial \hat{y}}{\partial z_2}\frac{\partial z_2}{\partial b_2}$$

2.2 2(2) Answer

Using the softplus function defined in the lecture slides:

$$\text{Softplus}(x) = \log(1 + \exp(x))$$

Given:

$$\sigma(z) = \text{Softplus}(z)$$

$$\hat{y} = \sigma(z_2)$$

$$\hat{y} = \log(1 + e^{z_2})$$

Calculating the derivative of the softplus function:

$$\frac{d}{dx}\text{Softplus}(x) = \frac{d}{dx}\log(1 + e^x)$$

$$\frac{d}{dx}\log(1 + e^x) = \frac{1}{1 + e^x} \cdot \frac{d}{dx}(1 + e^x)$$

$$\frac{1}{1+e^x} \cdot \frac{d}{dx}(1+e^x) = \frac{e^x}{1+e^x}$$
$$\frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$

Therefore, the derivative of the softplus function is the sigmoid function.

Calculating $\frac{\partial \hat{y}}{\partial z_2}$:

$$\frac{\partial \hat{y}}{\partial z_2} = \frac{\partial}{\partial z_2}\text{Softplus}(z_2) = \frac{1}{1+e^{-z_2}}$$
$$\frac{\partial z_2}{\partial b_2} = 1$$
$$\frac{\partial \hat{y}}{\partial b_2} = \frac{\partial \hat{y}}{\partial z_2}\frac{\partial z_2}{\partial b_2} = \frac{\partial \hat{y}}{\partial z_2} = \frac{1}{1+e^{-z_2}}$$

Calculating $\frac{\partial \hat{y}}{\partial w_2}$:

$$\frac{\partial z_2}{\partial w_2} = a_1$$
$$\frac{\partial \hat{y}}{\partial z_2} = \frac{\partial}{\partial z_2}\text{Softplus}(z_2) = \frac{1}{1+e^{-z_2}}$$
$$\frac{\partial \hat{y}}{\partial w_2} = \frac{\partial \hat{y}}{\partial z_2}\frac{\partial z_2}{\partial w_2} = \frac{a_1}{1+e^{-z_2}}$$

2.3 2(3) Answer

Given:

$$\sigma(z) = \text{Softplus}(z)$$

Calculating $\frac{\partial \hat{y}}{\partial x}$:

$$\frac{\partial \hat{y}}{\partial x} = \frac{\partial \hat{y}}{\partial z_2}\frac{\partial z_2}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial x}$$
$$\frac{\partial \hat{y}}{\partial z_2} = \frac{\partial}{\partial z_2}\text{Softplus}(z_2) = \frac{1}{1+e^{-z_2}}$$
$$\frac{\partial z_2}{\partial a_1} = w_2$$
$$\frac{\partial a_1}{\partial z_1} = \frac{\partial}{\partial z_1}\text{Softplus}(z_1) = \frac{1}{1+e^{-z_1}}$$
$$\frac{\partial z_1}{\partial x} = w_1$$
$$\frac{\partial \hat{y}}{\partial x} = \frac{w_2}{1+e^{-z_2}}\frac{w_1}{1+e^{-z_1}} = \frac{w_2}{1+e^{-(w_2x+b_2)}}\frac{w_1}{1+e^{-(w_1x+b_1)}}$$

Since $\frac{\partial \hat{y}}{\partial x}$ is dependent on the value of b_2 as shown above, **changing the value of b_2 will change** $\frac{\partial \hat{y}}{\partial x}$.

2.4 2(4) Answer

Given:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Calculating $\frac{\partial\sigma(z)}{\partial z}$:

$$\frac{\partial\sigma(z)}{\partial z} = \frac{\partial}{\partial z}(\frac{1}{1+e^{-z}})$$

$$\frac{\partial}{\partial z}(\frac{1}{1+e^{-z}}) = \frac{\partial}{\partial z}(1+e^{-z})^{-1}$$

Using chain rule:

$$\frac{\partial}{\partial z}(1+e^{-z})^{-1} = -(1+e^{-z})^{-2} \cdot \frac{\partial}{\partial z}(1+e^{-z})$$

$$-(1+e^{-z})^{-2} \cdot \frac{\partial}{\partial z}(1+e^{-z}) = -(1+e^{-z})^{-2} \cdot \frac{\partial}{\partial z}(e^{-z})$$

$$-(1+e^{-z})^{-2} \cdot \frac{\partial}{\partial z}(e^{-z}) = -(1+e^{-z})^{-2} \cdot (e^{-z} \cdot \frac{\partial}{\partial z}(-z))$$

$$-(1+e^{-z})^{-2} \cdot \frac{\partial}{\partial z}(e^{-z}) = -(1+e^{-z})^{-2} \cdot (e^{-z} \cdot \frac{\partial}{\partial z}(-z))$$

$$-(1+e^{-z})^{-2} \cdot (e^{-z} \cdot \frac{\partial}{\partial z}(-z)) = (1+e^{-z})^{-2} \cdot e^{-z}$$

Which can be rewritten as:

$$(1+e^{-z})^{-2} \cdot e^{-z} = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}}$$

$$\frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} = \frac{1}{1+e^{-z}} \cdot (\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}})$$

$$\frac{1}{1+e^{-z}} \cdot (\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}) = \frac{1}{1+e^{-z}} \cdot (1 - \frac{1}{1+e^{-z}})$$

$$\frac{1}{1+e^{-z}} \cdot (\frac{1+e^{-z}}{1+e^{-z}} - \frac{1}{1+e^{-z}}) = \frac{1}{1+e^{-z}} \cdot (1 - \frac{1}{1+e^{-z}})$$

$$\frac{1}{1+e^{-z}} \cdot (1 - \frac{1}{1+e^{-z}}) = \sigma(z) \cdot (1-\sigma(z))$$

Calculating $\frac{\partial\hat{y}}{\partial w_1}$:

$$\frac{\partial\hat{y}}{\partial w_1} = \frac{\partial\hat{y}}{\partial z_2}\frac{\partial z_2}{\partial a_1}\frac{\partial a_1}{\partial z_1}\frac{\partial z_1}{\partial w_1}$$

$$\frac{\partial\hat{y}}{\partial z_2} = \frac{\partial\sigma(z_2)}{\partial z_2} = \frac{1}{1+e^{-z_2}}(1 - \frac{1}{1+e^{-z_2}})$$

$$\frac{\partial z_2}{\partial a_1} = w_2$$

$$\frac{\partial a_1}{\partial z_1} = \frac{\partial\sigma(z_1)}{\partial z_1} = \frac{1}{1+e^{-z_1}}(1 - \frac{1}{1+e^{-z_1}})$$

$$\frac{\partial z_1}{\partial w_1} = x$$

$$\frac{\partial \hat{y}}{\partial w_1} = \frac{w_2}{1+e^{-z_2}}(1 - \frac{1}{1+e^{-z_2}})\frac{x}{1+e^{-z_1}}(1 - \frac{1}{1+e^{-z_1}})$$

or

$$\frac{\partial \hat{y}}{\partial w_1} = w_2 x \cdot \sigma(z_2)(1-\sigma(z_2)) \cdot \sigma(z_1)(1-\sigma(z_1))$$

3 Naive Bayes [35 points]

Suppose we have a dataset of individuals who have been audited by the IRS for tax evasion. The data includes three features/attributes: 'Refund' (yes or no), 'Marital Status' (single, married, divorced), and 'Taxable Income' (a continuous value). The target variable is whether or not the individual was found guilty of tax evasion (yes or no). The dataset is shown in the Table 1.

Note that for continuous attribute 'Taxable Income', we assume it follows a class-conditional normal distribution, which means $P(\text{Taxable Income}|\text{Evade Tax} = \text{c}) \quad N(\mu_c, \sigma_c^2)$ where $c \quad \{\text{Yes, No}\}$ is the value of output variable, i.e. 'Evade Tax'. Specifically, the probability density function of $N(\mu_c, \sigma_c^2)$ is $P(x|Y=c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} exp(-\frac{(x-{}_c)^2}{2\sigma_c^2})$. The sample mean μ_c is computed as $\mu_c = \frac{1}{n_c}\sum_{i=1}^{n_c} x_i$, where n_c is the number of samples w.r.t. class c in the training set. The sample variance σ_c^2 is computed as $\sigma_c^2 = \frac{1}{n_c-1}\sum_{i=1}^{n_c}(x_i - \mu_c)^2$, where having $(n_c - 1)$ instead of n_c in the denominator is because of the use of Bessel's correction.

Your task is to implement a Naive Bayes classifier to predict whether an individual is likely to evade taxes or not, based on his/her refund status (Yes), marital status (Married), and taxable income (79K). Please clearly present the steps that lead to your final predictions.

[ ]:
```
display(Image(filename="table_1.png"))
```

Refund	Marital Status	Taxable Income	Evade Tax
Yes	Single	125K	No
No	Married	100K	No
No	Single	70K	No
Yes	Married	120K	No
No	Divorced	95K	Yes
No	Married	60K	No
Yes	Divorced	220K	No
No	Single	85K	Yes
No	Married	75K	No
No	Single	90K	Yes

Table 1: The toy dataset of individuals who have been audited by the IRS for tax evasion.

3.1 Question 3 Answer

Calculate P(Evade Tax|Refund=Yes, Marital Status=Married, Taxable Income=79k), assuming that Laplace smoothing is not used (as per clarifications with Prof Zhao Na):

Evade Tax \ Refund	Yes	No
Yes	0	3
No	3	4

Evade Tax \ Marital Status	Single	Married	Divorced
Yes	2	0	1
No	2	4	1

Calculating the mean and standard deviation of the continuous variable, Taxable Income:

$\mu_{Yes} = \frac{1}{3}(95 + 85 + 90) \times 10^3 = 90 \times 10^3$

$\mu_{No} = \frac{1}{7}(125 + 100 + 70 + 120 + 60 + 220 + 75) \times 10^3 = 110 \times 10^3$

$\sigma^2_{Yes} = \frac{1}{2}[(95000 - 90000)^2 + (85000 - 90000)^2 + (90000 - 90000)^2] = 25 \times 10^6$

$\sigma^2_{No} = \frac{1}{6}[(125000 - 110000)^2 + (100000 - 110000)^2 + (70000 - 110000)^2 + (120000 - 110000)^2 + (60000 - 110000)^2 + (220000 - 110000)^2 + (75000 - 110000)^2] = 2975 \times 10^6$

P(Evade Tax=Yes|Refund=Yes, Marital Status=Married, Taxable Income=79k) = P(Evade Tax=Yes)×P(Refund=Yes|Evade Tax=Yes)×P(Marital Status=Married|Evade Tax=Yes)× P(Taxable Income=79k|Evade Tax=Yes)

P(Evade Tax=Yes) = $\frac{3}{10}$

P(Refund=Yes|Evade Tax=Yes) = $\frac{0}{3} = 0$

P(Marital Status=Married|Evade Tax=Yes) = $\frac{0}{3} = 0$

P(Taxable Income=79k|Evade Tax=Yes) = $\frac{1}{\sqrt{2\pi\sigma^2_{Yes}}} exp(-\frac{(79\times10^3-90\times10^3)^2}{2\sigma^2_{Yes}})$ = $\frac{1}{\sqrt{2\pi 25\times10^6}} exp(-\frac{(79\times10^3-90\times10^3)^2}{2\times25\times10^6}) = 7.0949 \times 10^{-6}$

P(Evade Tax=Yes|Refund=Yes, Marital Status=Married, Taxable Income=79k) = $\frac{3}{10} \times 0 \times 0 \times 7.0949 \times 10^{-6} = 0$

Alternatively, P(Evade Tax=No|Refund=Yes, Marital Status=Married, Taxable Income=79k) = P(Evade Tax=No)×P(Refund=Yes|Evade Tax=No)×P(Marital Status=Married|Evade Tax=No)× P(Taxable Income=79k|Evade Tax=No)

P(Evade Tax=No) $= \frac{7}{10}$

P(Refund=Yes|Evade Tax=No) $= \frac{3}{7} = 0$

P(Marital Status=Married|Evade Tax=No) $= \frac{4}{7} = 0$

P(Taxable Income=79k|Evade Tax=No) $= \frac{1}{\sqrt{2\pi\sigma^2_{No}}} exp(-\frac{(79\times10^3 - 110\times10^3)^2}{2\sigma^2_{No}}) = \frac{1}{\sqrt{2\pi\times2975\times10^6}} exp(-\frac{(79\times10^3-110\times10^3)^2}{2\times2975\times10^6}) = 6.223\times10^{-6}$

P(Evade Tax=No|Refund=Yes, Marital Status=Married, Taxable Income=79k) $= \frac{7}{10} \times \frac{3}{7} \times \frac{4}{7} \times 6.223\times10^{-6} = 1.07\times10^{-6}$

Since P(Evade Tax=No|Refund=Yes, Marital Status=Married, Taxable Income=79k) $>$ P(Evade Tax=Yes|Refund=Yes, Marital Status=Married, Taxable Income=79k), we predict that the individual is **not likely to evade taxes**.

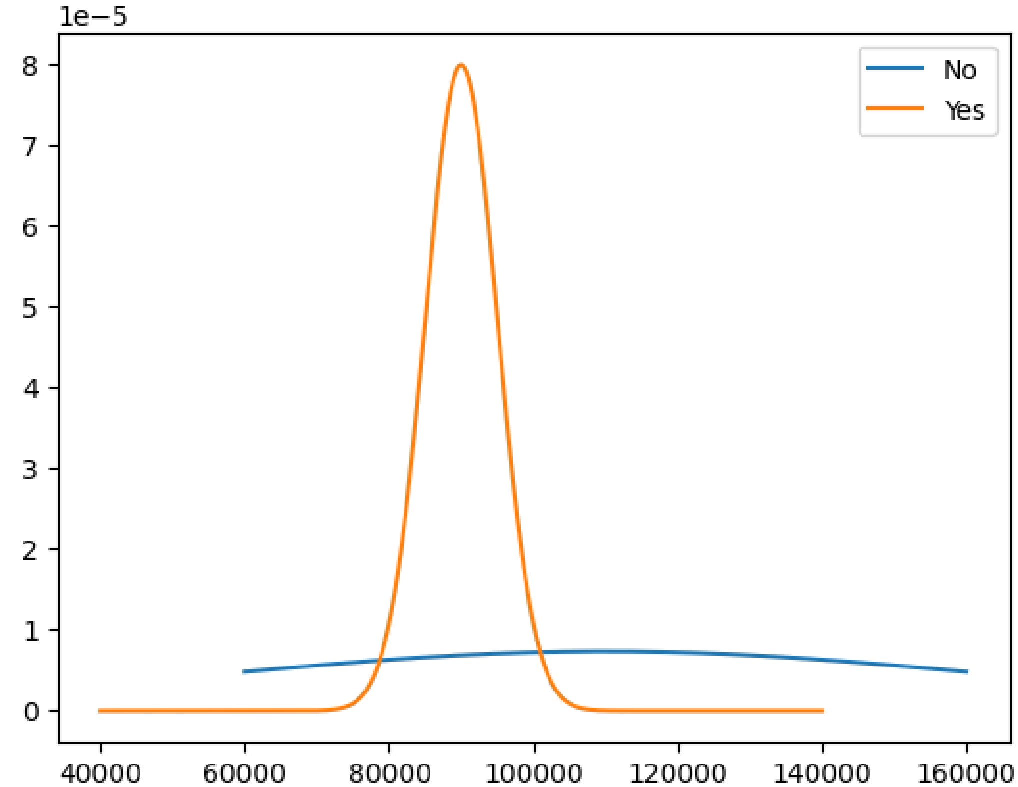
[ ]:
```
from scipy.stats import norm

range_from_mean = 50000

mean_no = 110000
sd_no = np.sqrt(2975 * 10**6)
x_axis_no = np.arange(-range_from_mean + mean_no, range_from_mean + mean_no, 1)

mean_yes = 90000
sd_yes = np.sqrt(25 * 10**6)
x_axis_yes = np.arange(-range_from_mean + mean_yes, range_from_mean + mean_yes,
  1)

plt.plot(x_axis_no, norm.pdf(x_axis_no, mean_no, sd_no), label="No")
plt.plot(x_axis_yes, norm.pdf(x_axis_yes, mean_yes, sd_yes), label="Yes")
plt.legend()
plt.show()
```



4 References

Prof Zhao Na

https://youtu.be/tIeHLnjs5U8

https://www.youtube.com/watch?v=Po6lsacF5pw

https://towardsdatascience.com/derivative-of-the-sigmoid-function-536880cf918e

https://neuralthreads.medium.com/softplus-function-smooth-approximation-of-the-relu-function-6a85f92a98e6