CIS 520, Machine Learning, Fall 2021
Homework 7
Due: Tuesday, November 2nd, 11:59pm
Submit to Gradescope

Yixuan Meng, Zhouyang Fang

1 Programming: Missing Data Imputation

For this question, refer to the Jupyter Notebook. You will be implementing different imputation techniques – the notebook will have detailed instructions.

1.1 Zero Imputation

Look for the code marked `#TODO` in the notebook and complete the code between the segments according to the instructions.

- Add the accuracy and the Frobenius norm in this report.

Frobenius Norm	51.668
Accuracy	60.0%

Table 1: Accuracy and Frobenius norm for Zero Imputation

- Add the code of the completed zeroImpute method.

```
X_imputed = np.nan_to_num(X_imputed)
```

Listing 1: zeroImpute method

1.2 Mean Imputation

Look for the code marked `#TODO` in the notebook and complete the code between the segments according to the instructions.

- Add the accuracy and the Frobenius norm in this report.

Frobenius Norm	13.76
Accuracy	84.0%

Table 2: Accuracy and Frobenius norm for Mean Imputation

- Add the code of the completed meanImpute method.

```
colmean = np.nanmean(X_imputed,axis = 0)
X_imputed = np.array([[colmean[n] if np.isnan(X_imputed[m][n]) \
        else X_imputed[m][n] for n in range(X_imputed.shape[1])] \
        for m in range(X_imputed.shape[0])])
```

Listing 2: meanImpute method

1.3 Regression Imputation

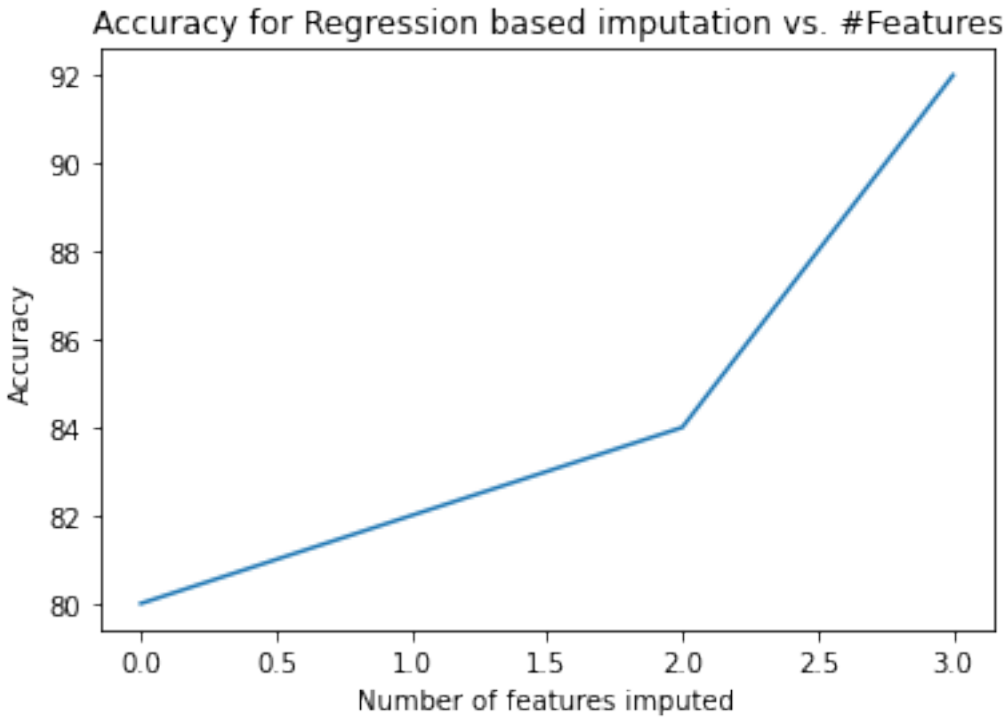
Look for the code marked `#TODO` in the notebook and complete the code between the segments according to the instructions.

- Complete the following table.

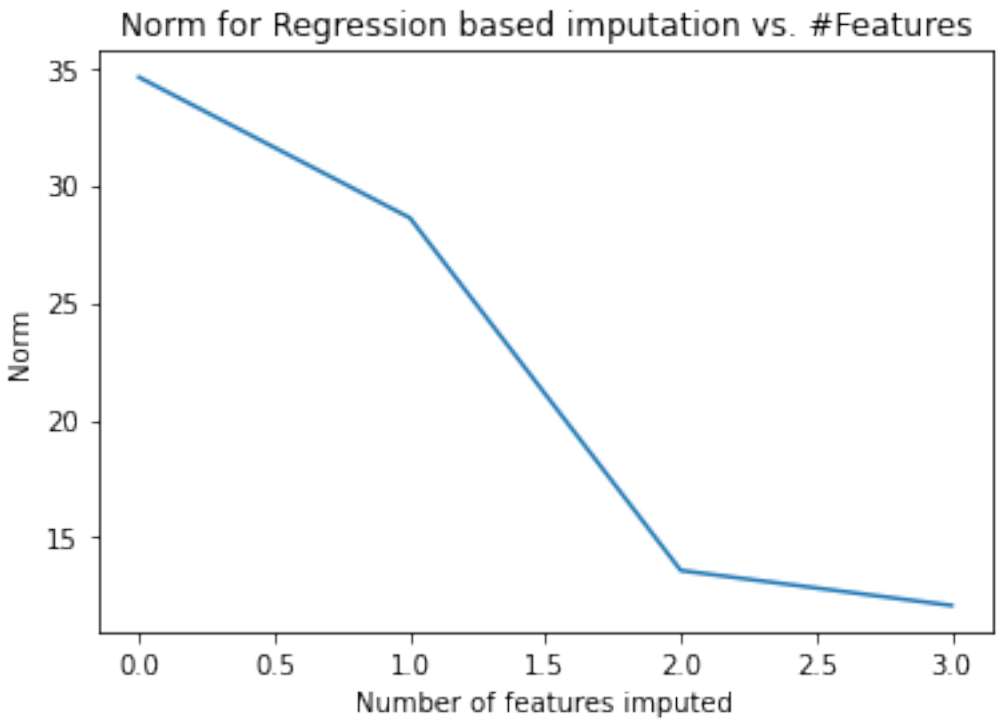
Epoch	Frobenius Norm	Accuracy
After Base Imputation	51.67	60%
1	12.10	92%
2	7.98	92%
3	8.13	96%
4	7.53	96%
5	7.83	98%

Table 3: Accuracy and Frobenius norm for Zero Imputation

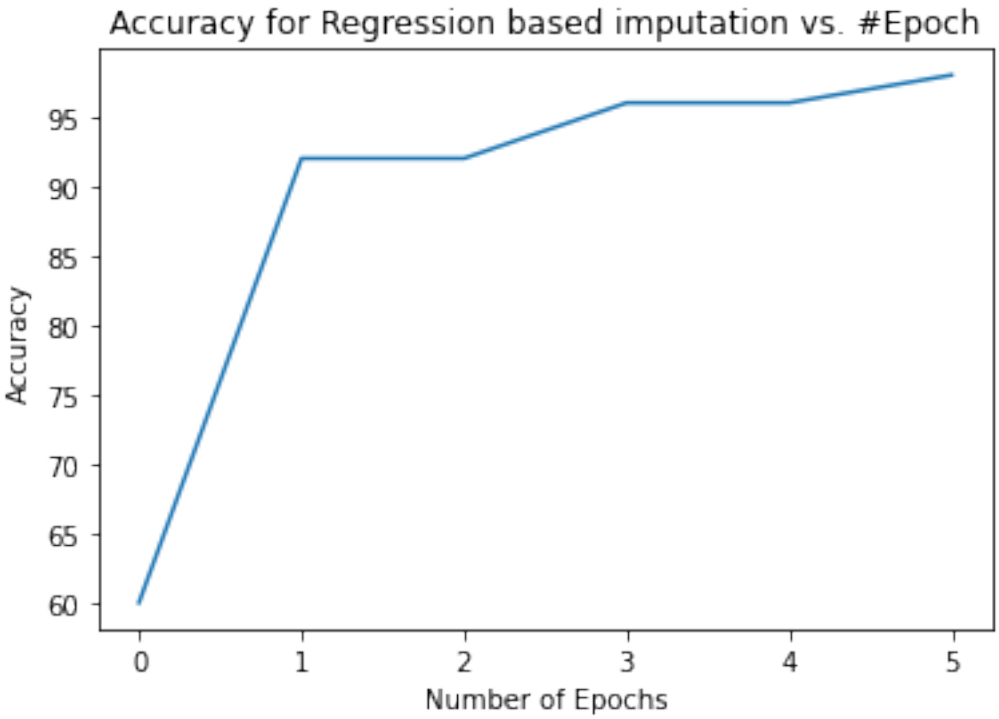
- Plot for Accuracy for Regression based imputation vs. Number of Features imputed



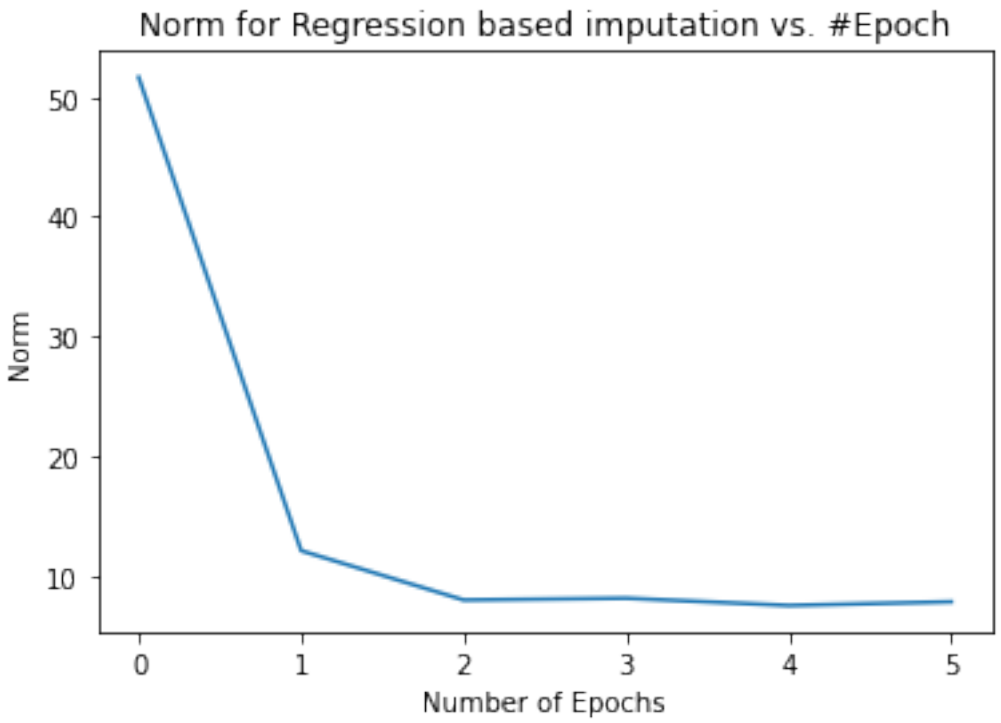
- Plot for Norm for Regression based imputation vs. Number of Features imputed



- Plot for Accuracy for Regression based imputation vs Number of Epochs



- Plot for Norm for Regression based imputation vs. Number of Epochs



- Add the code of the completed regressedImpute method.

```
m,n = X_imputed.shape
for col in range(n):
    nan_idx = np.where(np.isnan(X_miss[:,col]))[0]
    real_idx = np.where(np.isnan(X_miss[:,col])==False)[0] # rows which do not have
  missing data for the given column
    X_train_sub = X_baseImputed[real_idx, :]
    X_train_sub = np.delete(X_train_sub,col,1)
    y_train_sub = X_baseImputed[real_idx, col]

    clf = LinearRegression()
    clf.fit(X_train_sub, y_train_sub)
    X_val_sub = X_baseImputed[nan_idx,:]
    X_val_sub = np.delete(X_val_sub,col,1)
    new_y = clf.predict(X_val_sub)
    X_imputed[nan_idx,col] = new_y

    if computePerFeatureStatistics == True:
      # TODO 1.3.2
      clf = LogisticRegression()
      clf.fit(X_imputed,y_miss)
      frobenius_norms.append(LA.norm(X_imputed - X_train))
      accuracies.append(clf.score(X_test,y_test)*100)
```

Listing 3: regressedImpute method

1.4 Follow Up Questions

1. Which is the best of the three imputation methods? Why (briefly)?

   Regression Imputation. Frobenius norm for regression imputation is the smallest, and the logistic regression model trained on data imputed this way has the higheset accuracy.

2. Could you potentially do a better job of imputation if you also used y vales as well as x's? How might that help? If using the y's for imputation helps, why do people mostly not use them?

Hint: think through the whole process of model estimation and use.

Yes, adding y values as a feature will increase the logistic model built later since more features will probably helps to find the underlying distribution of data. We generally does not use y because we use x's to predict it and it should not be treated as known variable, using y's value in this way could cause overfitting.

3. Describe the trend for the accuracy and the norm as impute more features for regression imputation. Give a plausible explanation behind the trend.

   As the number of imputed features increases, the accuracy increases, the norm decreases. Because we replace more missing data with a reasonable value and provide more information for the regressor.

4. Describe the trend for the accuracy and the norm as re-impute again for several epochs for regression imputation. Give a plausible explanation behind the trend.

   As we do more re-impute, the accuracy increases, the norm decreases. The previous imputations are used when we re-impute the data, making the imputation better and the imputed data more similar to the real distribution.

2 Programming: K-Means Analysis

In this assignment, we will implement the K-means algorithm and perform it on a breast cancer data set. Please follow the attached Python notebook to write and plot the functions.

2.1 Part 1: K-means iteration function

Write the K-means iteration function as is described in the notebook. Each iteration takes the data and current centroid values for each class as parameters, and returns both updated centroid values along with a list of labels telling which class each datapoint belongs to.

1. 8 Fill in the table below by reporting the resulting cluster labels and resulting centroids from running the iteration function on the given parameters.

X	**Initial Centroids**	**Resulting Cluster Labels**	**Resulting Centroids**
[[4], [7], [11], [17]]	[1, 2]	[1, 1, 1, 1]	[[1], [9.75]]
[[4], [7], [11], [17]]	[1, 13]	[0, 0, 1, 1]	[[5.5], [14]]
[[4], [7], [11], [17]]	[18,19]	[0, 0, 0, 0]	[[9.75], [19]]
[[0,7,0],[4,5,0],[0,4,3],[2,3,4]]	[[2.5,4,0],[-2.5,5,0]]	[1, 0, 0, 0]	[[2 , 4, 2.333], [0, 7, 0]]

2. 2 If an iteration of the k-means algorithm returns less than K classes, what might that indicate about the data?

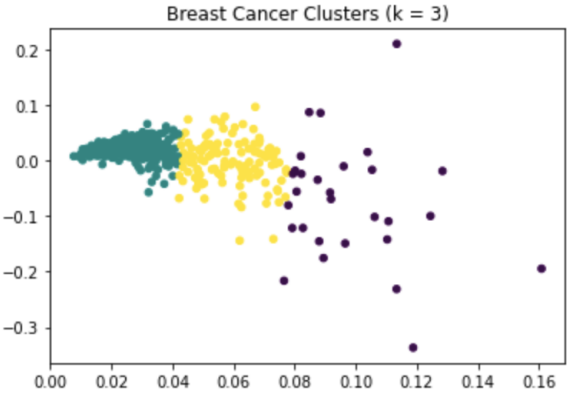
   We may use too many classes to represent the data. Few clusters compose this data pattern indeed.

2.2 Part 2: Putting the algorithm together

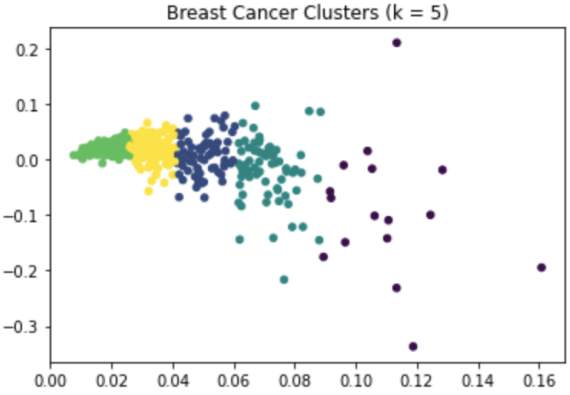
Now write the whole K-means function using your iteration function, as is described in the notebook. The function takes in the data set with the number of classes and returns the final centroid values, the final list of labels telling which class each data point belongs to, and the number of K-means iterations.

1. Put your 3 sanity-check graphs here.

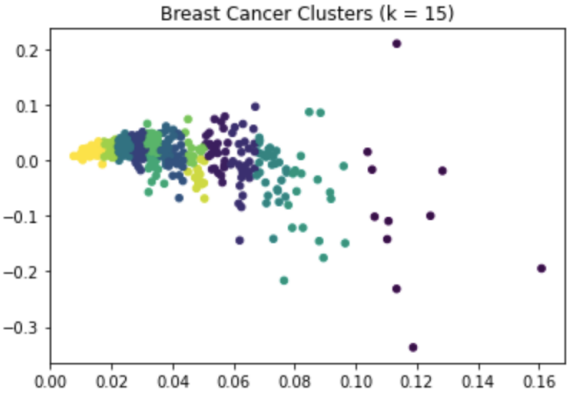
6



11



28



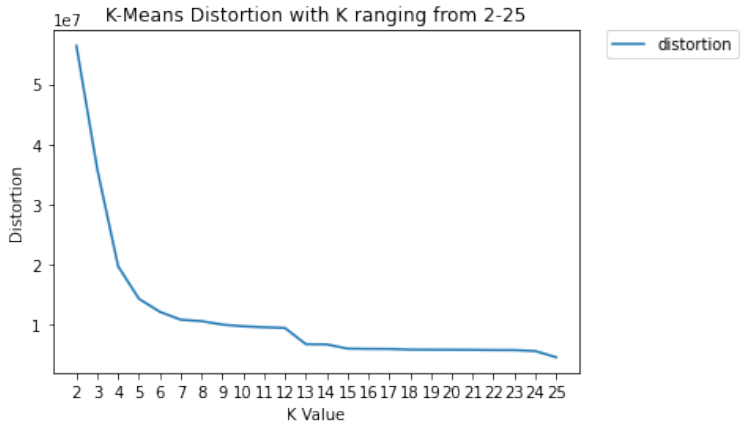
2. Write down how many iterations it took for your k-means to converge for each of the sanity-check graphs. How does this number compare with what you expected it to be? How does the number of iterations seem to vary proportional to k?

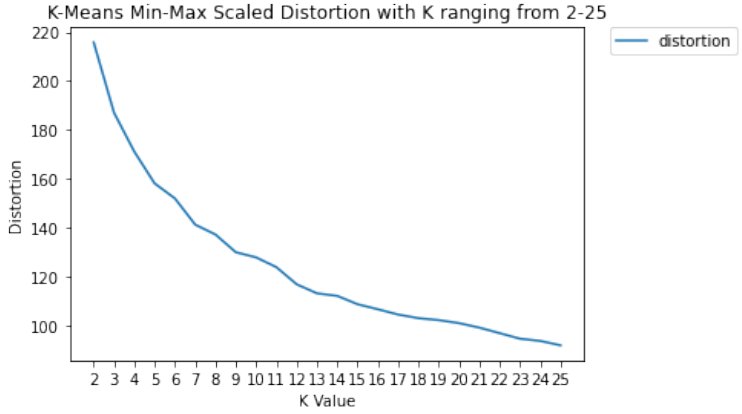
   6, 11, 28 iterations respectively. This number increases faster than expected. This number increases far slower than k.

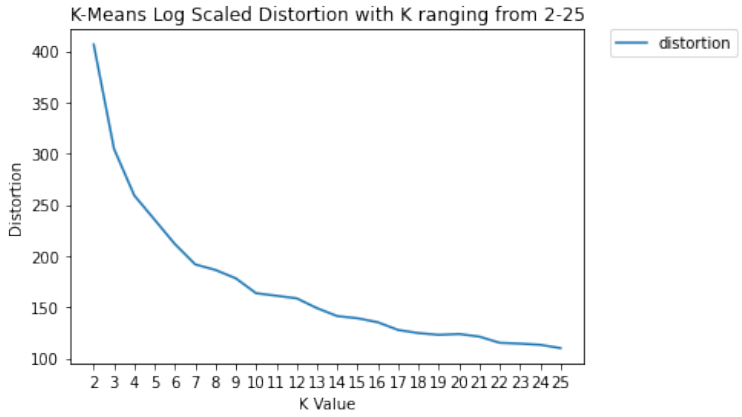
2.3 Part 3: Choosing the value of K

Write the test_cluster_size function using your k-means function, as is described in the notebook. The function takes in the data set with the number of classes and returns a list of scores where each score is the distortion for that value of k.

1. Using the breast cancer data set, record your labeled plots for distortion, min-max scaled distortion, and log scaled distortion here, ranging k from 2 to 25.







2. Why can't we hold out some of the data in a validation set, and choose the value of k which minimizes a cross-validation error like we have in the past for algorithms like linear regression?

   K-Means is an unsupervised method so there is no ground truth. Only supervised learning like linear regression has correct answers as well as cross-validation error.

3 EM Algorithm with Red and Blue Coins

Your friend has two coins: a red coin and a blue coin, with biases p_r and p_b, respectively (i.e. the red coin comes up heads with probability p_r, and the blue coin does so with probability p_b). She also has an inherent preference π for the red coin. She conducts a sequence of m coin tosses: for each toss, she first picks either the red coin with probability π or the blue coin with probability $1-\pi$, and then tosses the corresponding coin; the process for each toss is carried out independently of all other tosses. You don't know which coin was used on each toss; all you are told are the outcomes of the m tosses (heads or tails). In particular, for each toss i, define a random variable X_i as

$$X_i = \begin{cases} 1 & \text{if the } i\text{-th toss results in heads} \\ 0 & \text{otherwise.} \end{cases}$$

Then the data you see are the values $x_1, \ldots, x_m$ taken by these m random variables. Based on this data, you want to estimate the parameters $\theta = (\pi, p_r, p_b)$. To help with this, for each toss i, define a latent (unobserved) random variable Z_i as follows:

$$Z_i = \begin{cases} 1 & \text{if the } i\text{-th toss used the red coin} \\ 0 & \text{otherwise.} \end{cases}$$

1. Let X be a random variable denoting the outcome of a coin toss according to the process described above, and let Z be the corresponding latent random variable indicating which coin was used, also as described above (both X and Z take values in $\{0, 1\}$ as above). Write an expression for the joint distribution of X and Z. Give your answer in the form

$$p(x, z; \theta) = \left(\pi \left(p_r\right)^x \left(1-p_r\right)^{1-x}\right)^z \left(\left(1-\pi\right)\left(p_b\right)^x \left(1-p_b\right)^{1-x}\right)^{1-z}$$

.

2. Write an expression for the complete-data log-likelihood,

$$\begin{aligned} &\ln Ł_c(\theta) \\ =&\sum_{i=1}^{m} \ln p(x_i, z_i; \theta) \\ =&\sum_{i=1}^{m} \left(z_i \left(\ln \pi + x_i \ln\left(p_r\right) + \left(1-x_i\right)\ln\left(1-p_r\right)\right) + \left(1-z_i\right)\left(\ln(1-\pi) + x_i \ln\left(p_b\right) + \left(1-x_i\right)\ln\left(1-p_b\right)\right)\right). \end{aligned}$$

3. Suppose you knew the values z_i taken by the latent variables Z_i. What would be the maximum-likelihood parameter estimates $\hat{\theta}$? Give expressions for $\hat{\pi}$, $\hat{p_r}$, and $\hat{p_b}$ (in terms of x_i and z_i).

$$\begin{aligned} &\frac{\partial \ln Ł_c(\theta)}{\partial \pi} = 0 \\ \implies &\hat{\pi} = \frac{1}{m}\sum_{i=1}^{m} z_i \\ &\frac{\partial \ln Ł_c(\theta)}{\partial p_r} = 0 \\ \implies &\hat{p_r} = \frac{\sum_{i=1}^{m}\left(z_i x_i\right)}{\sum_{i=1}^{m} z_i} \\ &\frac{\partial \ln Ł_c(\theta)}{\partial p_b} = 0 \\ \implies &\hat{p_b} = \frac{\sum_{i=1}^{m}\left(1-z_i\right)x_i}{\sum_{i=1}^{m}\left(1-z_i\right)} \end{aligned}$$

4. In the absence of knowledge of z_i, one possibility for estimating θ is to use the EM algorithm. Recall that the algorithm starts with some initial parameter estimates θ^0, and then on each iteration t, performs an E-step followed by an M-step. Let θ^t denote the parameter estimates at the start of iteration t. In the E-step, for each toss i, the algorithm requires computing the posterior distribution of the latent variable Z_i under the current parameters θ^t. Calculate the posterior probability $P(Z_i = 1 \,|\, X_i = x_i;\, \theta^t)$. *(Hint: Use Bayes' rule.)*

$$\begin{aligned}
&P(Z_i = 1 \,|\, X_i = x_i;\, \theta^t) \\
=&\frac{p\left(X_i = x_i \mid Z_i = 1, \theta^t\right) p\left(Z_i = 1 \mid \theta^t\right)}{p\left(X_i = x_i \mid \theta^t\right)} \\
=&\frac{\pi^t \left(p_r^t\right)^{x_i} \left(1 - p_r^t\right)^{1-x_i}}{\pi^t \left(p_r^t\right)^{x_i} \left(1 - p_r^t\right)^{1-x_i} + \left(1 - \pi^t\right) \left(p_b^t\right)^{x_i} \left(1 - p_b^t\right)^{1-x_i}}
\end{aligned}$$

5. For each toss i, denote the posterior probability computed in part (d) above by γ_i^t (so that $\gamma_i^t = P(Z_i = 1 \,|\, X_i = x_i;\, \theta^t)$). Then the expected complete-data log-likelihood with respect to these posterior distributions is

$$\sum_{i=1}^{m} \left(\gamma_i^t \cdot \ln p(x_i, 1;\, \theta) + (1 - \gamma_i^t) \cdot \ln p(x_i, 0;\, \theta) \right).$$

The M-step of the EM algorithm requires finding parameters θ^{t+1} that maximize this expected complete-data log-likelihood.

Determine the updated parameters θ^{t+1}. Give expressions for π^{t+1}, p_r^{t+1}, and p_b^{t+1} (in terms of x_i and γ_i^t).

Replacing z_i with γ_i^t, we get

$$\begin{aligned}
\hat{\pi}^{t+1} &= \frac{\sum_{i=1}^{m} \gamma_i^t}{m} \\
\hat{p}_r^{t+1} &= \frac{\sum_{i=1}^{m} \left(\gamma_i^t x_i\right)}{\sum_{i=1}^{m} \gamma_i^t} \\
\hat{p}_b^{t+1} &= \frac{\sum_{i=1}^{m} \left(\left(1 - \gamma_i^t\right) x_i\right)}{\sum_{i=1}^{m} \left(1 - \gamma_i^t\right)}.
\end{aligned}$$