I did not collaborate with anyone on this assignment.

Part 2: writing

2.1: Experiment design

Impurity measure: For my impurity measure, I chose the Gini index because for Random Forest, that is the default measure. Gini index is useful in this case because it is faster to compute than entropy, it has comparable performance to entropy, and Gini index pairs nicely with random forest.

Attribute splitting strategy: This is handled automatically by the library because we are using scikit-learn's random forest. For categorical attributes, we convert them to one-hot vectors, so each new binary column is treated as numeric, then random forest finds the best threshold splits. For continuous attributes, the library tries different numeric thresholds for each feature, and it finds the threshold that best reduces the gini impurity.

Data structure choice: We use a pandas dataframe for preprocessing, then we drop 'person ID' and the target column to get our matrix, and we store the target column as a numeric vector y. We use a random forest model, which is an ensemble of decision trees. Then for the para files, we use the para2_files to store the training ID's, the para3_files for the testing ID's then the para4_files to store our results.

Additional techniques: I gender mapped gender from 'm'/'f' to 1/0 to make it easier.

2.2: Model evaluations

The model evaluation scenario I chose was hold-out. I use the para2_file to list the people I will be using for the training data. Then I use the para3_file to get a different set of people, the test set, where I use the model created from the previous set and test it on the test set. Then the para4_file gives the prediction, which we can validate by going back to the original dataset. I chose this approach because I believed it would give accurate predictions. For my model evaluation metrics, I received a 0.8132% for accuracy, 0.800% for precision, 0.88% for recall, 0.8381 for my F1-score, and a 0.88 for my specificity (sensitivity is the same as recall). At first, I tried implementing a custom decision tree based on entropy, but my accuracy would be between 55% - 63%. Then I switched to random forest with the gini index, and this model proved to be significantly more reliable, and I believe it was for how the attributes were handled. I was able to effectively find the best thresholds to split, and effectively pre-process the data.

Part 3: Clustering Algorithms.

1. k-means++ :

```
Input: Dataset X, # of clusters k
1. Choose one center $c_1$ uniformly at random from Dataset X
2. For each point x in Dataset X, compute $D(x)$, which is the distance
to the nearest chosen center
3. Choose another center $c_i$ from Dataset X with probability proportional to
$D(x)^2$
4. Repeat steps 2 and 3 until there are k - centers
5. Proceed with standad k-means:
      - Repeat until convergence
      a. Assign each point to nearest center
      b. Update each center to be the mean of the cluster
```

One limitation of this method is that it is sensitive to outliers, and it assumes spherical clusters. But this method improves initialization by spreading out the initial centroids to get a more stable convergence.

2. k-medoids

```
Input: Dataset x, # of clusters k
1. Initialize: Randomly select k points from X as initial medoids
2. Repeat until convergence:
   a. Assign each point to the nearest medoid (based on distance)
   b. For each cluster:
        i. Try swapping medoid with a non-medoid point
        ii. Compute the total cost (sum of distances)
        iii. If total cost is reduced, perform the swap
```

This method is computationally expensive, and some versions don't scale well. But this method is more robust to outliers, and it's suitable for data where you can't compute the mean, like categorical data.