Data Preparation

In preparing our data, we first **identified the shape and size** of the raw data to get an understanding of how many rows and columns there are.

Thereafter, we wanted to get information about the **datatypes** of each column to determine whether to **perform conversions**. After running *data.info()*, we saw that all columns except for one had object Dtype. This meant that we had to look at some of the data content to determine whether a column had numerical or categorical values.

In doing so, we were able to tell whether to convert a column to numerical (float) or one-hot encode. Notably, many of the columns with object Dtype had numerical ranges (i.e. 30-34, 15-19 etc.).

There were two approaches I was considering: (1) take the midpoint of the range and set it as type float, or (2) treat them as categorical and one-hot encode them. I went with option (2) in order to preserve the aspect of the ranges present in the original data and to maintain interpretability.

The rest of the object Dtype columns were clearly categorical, so we performed one-hot encoding for them as well.

Moving forward, we had to **identify missing data, invalid values, and duplicated rows/values** and address them accordingly. Fortunately, there were no null values present in the dataset. As for duplicated rows, we identified 14 duplicated rows and thus dropped them. After doing so, it was important to check the *data.shape* to keep in mind the updated amount of rows.

Thereafter, we checked for **invalid values** across the dataset. With the help of chatGPT, I was able to write a method that outputs the rows with invalid values, allowing me to see what type of invalid values there were. From code block [10], we saw that ‘?’ was the only invalid value, so we simply replaced all of the ‘?’ occurrences with NaN.

Thereafter, we had to **replace these NaN’s with a suitable statistic**. For the one numerical column, *deg-malig,* we didn’t have to replace anything as there were no invalid/NaN values according to blocks [13] and [14]. As for the categorical columns, we replace the NaN values with the column’s mode.

After performing such tasks to address missing data, invalid values, and duplicated rows/values, it was important to validate our work by checking *data.info*, *check_invalid_values(data),* and whether ‘?’ was present anywhere in the dataset.

Finally, we had to **perform one-hot encoding** on the categorical variables in order to create our models.

Training Model

In training the three models, we had to use important techniques to prepare for training the model.

The first important task was to split the dataset into training and testing accordingly. Taking into account the fact that our dataset only contained 217 rows, which is not many, we dedicated at least 70% to the training set. We tested 70/30, 75/25, and 80/20 across all models, choosing the best split that produced optimal recall scores.

For our first model, K-Nearest Neighbor, we performed cross-validation, an important technique that helps identify the top performing *n_neighbors* value. Essentially, we train many different KNN models for different values of k and see which one performs best. Doing so allows us to improve accuracy.

Each Model's Performance to Predict Recurrence

Model	**Accuracy**
Knn	0.72
Decision tree	0.66
Naive Bayes	0.72

We can see that the Knn and Naive Bayes have the same accuracy, whereas the decision tree accuracy is the lowest. This attributes to the weaknesses of decision trees - that they are relatively poor predictors, prone to overfitting, sensitive to data fluctuations etc.

Model Recommendation

In the context of our project, we aim to minimize false negatives (improve recall), because it would be terrible to diagnose someone to not have recurrence of breast cancer, when in actuality they did.

Thus, from the scores, the decision tree model had the best recall score, hence why we would recommend this.