## Method summary:

A Decision tree classifier was chosen as one of the models to train on and validate the performance of the imputation techniques. We used the DecisionTreeClassifier model from the sklearn package in python, which applies an optimized CART tree.

This implementation required the input data to be numeric, so we encoded the categorical input features into integer values during preprocessing.

We used the test and train datasets (which contain no imputed missing values) to empirically determine the optimal tree\_depth hyperparameter. This was done by comparing the training and testing F1 scores at various values for tree\_depth. The optimal value is normally chosen at the point where the testing accuracy starts to decrease (indicating overfitting and loss of generalization). In our case there was no obvious turning point, and we instead chose the point where the testing score stopped improving.

We then trained the model (Now with optimal hyperparameters) on the various datasets and tested against the non-imputed base table. For each dataset we recorded the accuracy and F1 score, which was then used to compare model performance across different datasets.

We trained on the base dataset (no missing values induced, or missing values imputed) and report the accuracy and F1 score for this model as the baseline values. These baseline values are then compared to the paired imputed datasets (Same % of missing values imputed using Mode value and Naïve Bayes).

## Findings and observations:

It is important to note that the prediction target was a heavily skewed multiclass problem. When reporting accuracy and F1 scores, this is a weighted average across the 5 potential prediction classes. This weight is based on the proportion of values in each class, i.e. larger classes will contribute more to the score. The individual prediction class accuracy and F1 can be very different compared to the overall value quoted. Whether this is a concern is very problem-specific, but for our uses the weighted average is suited to validating imputation techniques.

As can be seen by our graphs, both techniques performs very well at 10% missing values, and performance degrades as more missing values are introduced and imputed. Surprisingly, the classifier still predicted with remarkable accuracy at 70% missing values imputed. I suspect this is because the prediction class was skewed and missing values were introduced randomly, which means that on average the skewed class distributions would be maintained, and mode imputation would still guess the correct prediction class quite often.

This means that our results here might not be reproducible if we have non-random missing values or more even class distributions.

The mode imputation approach performs very similarly to the Naïve Bayes approach at all levels of missing values tested (Accuracy and F1 score within 3% difference). Naïve bayes does perform slightly better at higher levels of missing values, but not to such a degree that it would necessitate using the more involved approach.

As follow-up research I would propose running the experiment again with non-random missing values, and testing against a more balanced dataset.