Project 2: Breast Cancer Recurrence

By: Antonio Jimenez

Data Exploration:

Before preparing the data for our classification models it is important to know information about the data that you are working with. From analyzing the shape of the database using .shape, I found the data set to be 286 rows and 10 columns. I followed by using the .head() function and .info() function to evaluate what the data looked like, the data type, and the number of non-null values the database contained. From the .info() output I found that there are no null values and that a lot of the columns are object type. Upon investigating the unique values of each column, I discovered the columns "node-caps" and "breast-quad" have values labeled as "?" for missing data.

Data Preparation:

My first objective was to transform the "age", "tumor-size", and "inv-nodes" data from objects to float values. Given these columns were integer ranges, I created a function to replace the integer ranges with the mean value, which I then converted into a float. I then deleted the rows with missing data as the missing data was categorical. I proceeded to eliminate duplicate rows, which resulted in there being 263 rows left. This was done as the number of duplicate rows was relatively small, and the duplicate data varying largely, meaning it would not attribute largely to a bias in the data that we would want to be aware of. Lastly, I performed one-hot encoding on the "class", "menopause", "node-caps", "breast", "breast-quad", and "irradiat" columns due to their categorical nature.

Uni and Multivariate Plots:

Having prepared the data I used describe() to analyze if there were any outliers in the numerical data. While there were outliers, there was nothing that was illogical and worth eliminating. The plot generated by cell 30 shows that there were a lot of large outliers in the "inv-nodes" data. The plot in cell 31 shows us that most of the tumor-size ranges from 20 to 35mm with a few outliers. The plot in cell 32 helps us realize the age of patients follows a standard distribution with a mean around the age of 50. We also see that we have the most data for when the degree of malignancy is 2 based on the plot generated by cell 33. Additionally, I used a heat map, cell 35, to show the correlation between the columns. The most important information seen is that recurrence is slightly positively correlated to invasive nodes, tumor size, degree of malignancy, node capsule, and irradiation. I also found interesting the large correlation between node capsule and invasive nodes.

Building the Models:

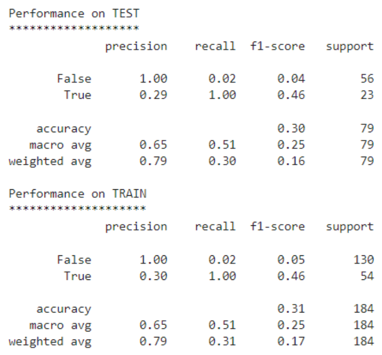
Having analyzed and prepared the data I performed a 70/30 train-test split with stratify applied to my dependent variable. For this project I selected the "class-recurrence-event" column to be my dependent variable as the purpose of this model is to try to predict which patients will have recurrence. I then selected the rest of the columns to be the independent variables. Using the split data I proceeded to creating a Random Forest classifier, K-Nearest Neighbor classifier(KNN), and a Multinomial Naïve Bayes classifier. Each of the models was created to optimize for recall as we are looking to minimize the number of false negatives. Because of this, when analyzing the performance of the models this report focuses on recall and accuracy. Having a large number of false negatives would mean that we would be classifying people as not being likely to have a recurrence event when they are likely to, which we want to minimize as we are dealing with breast cancer. Additionally, when finding the model that maximizes recall the number of folds was set to 5 when performing the grid search using GridSearchCV() from sklearn.

Random Forest Classifier:

To create the random forest classifier, I used the sklearn model and used a parameter grid to search for the model that performed the best in maximizing recall. The 4 parameters used in the search grid were number of estimators, max depth, min_samples_leaf, and class weight. Below is the parameter grid used:

```
param_grid = {
  "n_estimators": np.arange(start=4, stop=100, step=2),
  "max_depth": np.arange(start=2, stop=20),
  "min_samples_leaf": np.arange(start=1, stop=5),
  "class_weight": [{0: 0.1, 1: 0.9}, {0: 0.2, 1: 0.8}, {0: 0.3, 1: 0.7}],
}
```

The grid search resulted in “class_weight” = {0: 0.1, 1: 0.9}, “max_depth” = 2, “min_samples_leaf” =1, and “n_estimators” = 48. In figure 1 we can see the models performance on the test and training data. We see that the recall for our True values was equal to 1, but the accuracy on the test data was 0.3 which is quite poor. On the bright side, a recall value of 1 means that we had 0 cases of false negatives. Based on the results, it seems that the model is overfitting for maximizing recall resulting in poor accuracy.

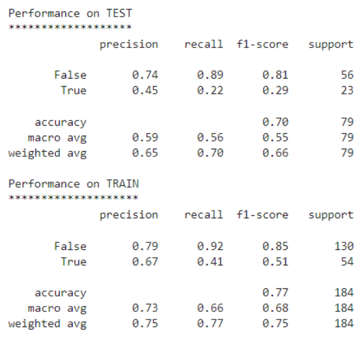


```
Performance on TEST
*******************
              precision    recall  f1-score   support

       False       1.00      0.02      0.04        56
        True       0.29      1.00      0.46        23

    accuracy                           0.30        79
   macro avg       0.65      0.51      0.25        79
weighted avg       0.79      0.30      0.16        79

Performance on TRAIN
********************
              precision    recall  f1-score   support

       False       1.00      0.02      0.05       130
        True       0.30      1.00      0.46        54

    accuracy                           0.31       184
   macro avg       0.65      0.51      0.25       184
weighted avg       0.79      0.31      0.17       184
```

Figure 1: Random Forest Classifier

K-Nearest Neighbor (KNN):

Similarly to the random forest classifier, I used sklearn’s KNeighborsClassifier() function to build the KNN. When optimizing for the recall I simply varied the hyperparameter “n-neighbors” using grid search. The grid search found the best value for “n-neighbors” to be 1, which would result in overfitting. When analyzing the results we see that the training accuracy is .97 and the test accuracy is .68, which further shows that the model is overfitting. I altered the range for the grid search to search from 5 to 100. As seen in Figure 2, we can see that the range between training and test accuracy largely decreased when the grid found 5 to be the optimal “n-neighbors” value. Furthermore, the model recall is 0.22 and the accuracy of the model is 0.7.

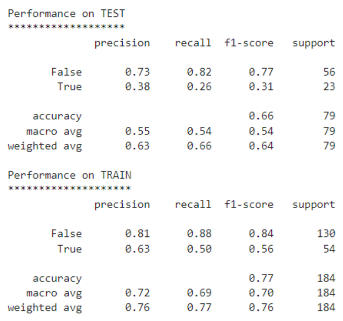


```
Performance on TEST
*******************
              precision    recall  f1-score   support

       False       0.74      0.89      0.81        56
        True       0.45      0.22      0.29        23

    accuracy                           0.70        79
   macro avg       0.59      0.56      0.55        79
weighted avg       0.65      0.70      0.66        79

Performance on TRAIN
********************
              precision    recall  f1-score   support

       False       0.79      0.92      0.85       130
        True       0.67      0.41      0.51        54

    accuracy                           0.77       184
   macro avg       0.73      0.66      0.68       184
weighted avg       0.75      0.77      0.75       184
```

Figure 2: KNN model performance.

Multinomial Naïve Bayes:

For the Naïve Bayes model I selected to use the Multinomial Naïve Bayes model as a lot of our data that is correlated with recurrence is discrete values. For this model, a grid search over the alpha hyperparameter was set which resulted in an alpha value of 1e-5 from the following values [0.00001, 0.0001, 0.001, 0.1, 1, 10, 100,1000]. As seen in Figure 3, the model had an accuracy of 0.66 and a recall of 0.26.



```
Performance on TEST
*******************
              precision    recall  f1-score   support

       False       0.73      0.82      0.77        56
        True       0.38      0.26      0.31        23

    accuracy                           0.66        79
   macro avg       0.55      0.54      0.54        79
weighted avg       0.63      0.66      0.64        79

Performance on TRAIN
********************
              precision    recall  f1-score   support

       False       0.81      0.88      0.84       130
        True       0.63      0.50      0.56        54

    accuracy                           0.77       184
   macro avg       0.72      0.69      0.70       184
weighted avg       0.76      0.77      0.76       184
```

Figure 3: Multinomial Naïve Bayes performance.

Further Optimization:

To further optimize the recall, we can alter the threshold for what is considered a True or likely to be a recurrence event which alters the recall and precision. In our case, a lower threshold means higher recall and lower precision. Given a range of thresholds the function find the threshold value that maximizes recall. A threshold of .1127 was applied to the Multinomial Naïve Bayes model from above (threshold = 0.5), which resulted in a recall of 0.57 and a precision of 0.43 for the test data.

Conclusion:

In conclusion the Random Forest model had the largest recall, but an extremely poor accuracy compared to the other models. KNN and Multinomial Naïve Bayes behaved similarly with KNN having slightly higher accuracy and lower recall than the Multinomial Naïve Bayes model. If my goal was to ensure most cases of recurrence were detected I would use the Random Forest model as it had the largest recall value. However, there would be a lot of false positives, which would result in increased cost to the patients who would have been non-recurrent. Additionally, the random forest model had the best F1 score indicating strong predictive power. Lastly, I used ChatGPT when figuring out the best practices to handle binned data.