PROJECT 1 REPORT

Breast Cancer Predictor

Part 1

In regards to the raw data, the shape is (386, 10), the size is 3860. This means 386 rows and 10 columns, with 3860 total entries. It doesn't appear that any of the variables need data type conversion, so none was performed. Tumor-size and inv-nodes have null values, which need to be treated. These were filled in with modes after grouping by age, as research online shows that involved nodes and tumor size can vary by/has a correlation to a person's age. Now there are no more null values.

Now, I check for duplicate rows and I notice that there are 11 duplicates. I then drop duplicates in place while reindexing the dataframe. Built-in Pandas methods come in very handy here. Then, before one-hot encoding, I perform univariate analysis on each of my columns and visualize them according to their frequency/how often they occur. This appears as a seaborn count plot. Most people in the dataset have no recurrence events, are middle-aged, are pre-menopause, have a tumor size around 30, have 0-2 involved nodes, no node caps, about equal in which breast it occurs in, mostly in the lower left breast quadrant, and have no irradiation. Knowing this information could help me remove bias within the dataset.

All columns are objects, these will need one hot encoding. Except for deg-malig which is an int64. In order to perform one hot encoding, I first convert every column except deg-malig to categorical data under Pandas. Now there are no more object data types. Now, using pd.get_dummies, I encoded all the categorical data into boolean data types.

Part 2

The data was split into X_train, X_test, y_train, y_test using train_test_split, with a 70/30 train-test ratio, and a y stratification to maintain class proportion. Below are the results of each model with different optimizations. The most important metric is recall.

Classification Report KNN_GSCV on TEST:

	precision	recall	f1-score	support
0	0.68	0.99	0.80	77
1	0.00	0.00	0.00	36
accuracy			0.67	113
macro avg	0.34	0.49	0.40	113
weighted avg	0.46	0.67	0.55	113

Classification Report KNN_GSCV on TRAIN:

	precision	recall	f1-score	support
0	0.68	0.99	0.80	177
1	0.33	0.01	0.02	85
accuracy			0.67	262
macro avg	0.50	0.50	0.41	262
weighted avg	0.56	0.67	0.55	262

KNN with GS, n=44

Fig 1

Classification Report KNN on TEST:

	precision	recall	f1-score	support
0	0.69	0.79	0.74	77
1	0.36	0.25	0.30	36
accuracy			0.62	113
macro avg	0.53	0.52	0.52	113
weighted avg	0.59	0.62	0.60	113

Classification Report KNN on TRAIN:

	precision	recall	f1-score	support
0	0.81	0.91	0.86	177
1	0.75	0.55	0.64	85
accuracy			0.79	262
macro avg	0.78	0.73	0.75	262
weighted avg	0.79	0.79	0.78	262

KNN w/o GS, n=3

Fig 2

Classification Report SGD Perceptron on TEST:

	precision	recall	f1-score	support
0	0.68	0.99	0.81	77
1	0.50	0.03	0.05	36
accuracy			0.68	113
macro avg	0.59	0.51	0.43	113
weighted avg	0.63	0.68	0.57	113

Classification Report SGD Perceptron on TRAIN:

	precision	recall	f1-score	support
0	0.70	0.99	0.82	177
1	0.89	0.09	0.17	85
accuracy			0.70	262
macro avg	0.79	0.54	0.49	262
weighted avg	0.76	0.70	0.61	262

Linear SGD Perceptron Loss

Fig 3

Classification Report KNN_GSCV_RECALL on TEST:

	precision	recall	f1-score	support
0	0.69	0.69	0.69	77
1	0.33	0.33	0.33	36
accuracy			0.58	113
macro avg	0.51	0.51	0.51	113
weighted avg	0.58	0.58	0.58	113

Classification Report KNN_GSCV_RECALL on TRAIN:

	precision	recall	f1-score	support
0	0.99	0.98	0.99	177
1	0.97	0.99	0.98	85
accuracy			0.98	262
macro avg	0.98	0.99	0.98	262
weighted avg	0.98	0.98	0.98	262

KNN with GS Op for Recall, n=1

Fig 4

Part 3

The steps taken to prepare the data are described in part one. I learned about where most of the data lies using count plots, also discussed in part one, and I realized that most of the data is categorical and needs a classification model for prediction. I used a total of four different methods to implement a predictive model. First I began with a linear classifier model using Scikit-Learn's stochastic gradient descent algorithm paired with the perceptron loss function **(Fig 3)**. This model attempts to draw a line of best fit through all the data points created by reducing the least squares cost function, then uses it to predict a value on the test set. The linear model has okay precision and is failing

recall and f1 score when trying to predict the recurrence of breast cancer. It performs well across all metrics when predicting the non-recurrence of breast cancer, and it evaluates to an overall accuracy of 68%.

For this dataset, the most important metric is recall. This is because we don't want to give patients a sense of false safety and tell them that they don't have breast cancer. It is okay to have false positives, aka a low precision, because it is better to err on the side of caution in terms of breast cancer. Since the linear model has very low recall when predicting recurrence, it is not a good fit.

Then, I switched over to using the K-Nearest Neighbor classifier. Initially I used 3 neighbors, which improved all metrics when predicting recurrence (**Fig 2**), however decreased in accuracy to 62%.

To try another method, I utilized KNN again but with grid search to implement cross validation and find the best hyperparameters for the model (**Fig 1**). Doing this significantly improved all metrics, especially recall when predicting no-recurrence, however completely fails all metrics when trying to predict recurrence. But it slightly improved its accuracy to 67%.

This model went all the way to the other corner in terms of what I want to optimize, so I then tried KNN with Grid Search, but optimized for recall (**Fig 4**). This model improved accuracy, recall, and the f1 score overall when predicting recurrence, and remained about the same when predicting no-recurrence. Its accuracy was the highest at 98%.

Overall, the model performance is acceptable for a simple binary classification algorithm. However I would not recommend using any of these models within a clinical setting, but if one had to be chosen, I would recommend the KNN model trained with grid search and optimized for recall.