Project 1 Report

Aastha Agrawal
aa92838

Data Preparation

I started by loading the dataset from the file **project1.csv**. My first step was to understand the size and shape of the data, so I used simple commands to print out the number of rows and columns. This helped me get a quick snapshot of the raw data and its structure.

After that, I dug a little deeper to check the data types of each column with a tool that tells you if numbers are stored as strings or if any columns need conversion. I also looked for duplicate rows, because having duplicates can sometimes skew your results. Luckily, if there were any duplicates, I made sure to remove them so that every data point would count only once.

Next, I turned my attention to missing or invalid data. I noticed that the dataset contained placeholders like "?" and "*" to represent missing values. To handle these correctly, I replaced them with a standard missing value marker (`np.nan`). Then, for each column with missing data, I decided on a simple strategy for imputation. For the categorical variables such as age, menopause, tumor-size, and others, I filled in the missing values with the mode—the value that appears most often. For the numeric column (deg-malig), I first made sure it was treated as a number and then filled any gaps with the median value. This step ensured that my dataset was complete and that no entry was left out because of missing information.

Finally, because machine learning models usually need numerical inputs, I transformed all the categorical features (except the target variable “class”) into a numerical format using one-hot encoding. This created new binary columns for each category, which means the model could easily work with the data without worrying about text strings.

Insights from Data Preparation

As I worked through these cleaning steps, a few interesting things popped out. For example, the distribution of the deg-malig column—representing the degree of malignancy—was spread out nicely, which confirmed that using the median for imputation made sense. The count plots for the categorical variables gave me a clear picture of how many patients belonged to each category. I could see patterns like which age groups were most common and how the recurrence events were distributed among different categories.

This hands-on data cleaning made it clear that a careful balance exists between the classes of recurrence events and non-recurrence events. I learned that even small inconsistencies or missing values can lead to unreliable predictions later on. In short, taking the time to clean the data really paid off because it set the stage for building a robust model.

Model Training Procedure

Once I had a clean and complete dataset, I moved on to building the prediction models. The first thing I did was split the data into two parts: a training set and a test set. I used a method that ensures the split is reproducible (by setting a random seed) and that the proportions of recurrence events and non-recurrence events are similar in both sets. This stratification is essential for making sure the model learns from a balanced sample.

I then trained three different types of models:

1. **Basic K-Nearest Neighbors (KNN) Classifier:**
   I started with a simple KNN model as a baseline. This model makes predictions based on the closest data points in the feature space. After training on the training set, I evaluated it using a detailed report that showed accuracy, precision, recall, and F1-score.
2. **Optimized KNN with Grid Search CV:**
   To see if I could improve on the basic KNN model, I used GridSearchCV to experiment with different numbers of neighbors and different weighting options (whether to weight all neighbors equally or to give more importance to closer neighbors). This tuning process helped me find the best combination of hyperparameters, which improved the overall performance of the KNN model.
3. **Logistic Regression (Linear Classification Model):**
   I also built a logistic regression model. This model is great for binary classification problems like this one because it predicts the probability of an event occurring. Logistic regression is often used as a baseline because it is simple, interpretable, and tends to perform well when the relationship between features and the target is roughly linear.

Model Performance and Confidence

When I evaluated the models, I looked at several performance metrics. Each model's classification report detailed its accuracy, precision, recall, and F1-score. In the context of predicting breast cancer recurrence, recall is especially important. In a medical setting, it's crucial not to miss patients who might experience a recurrence. Even if this means that some patients are flagged unnecessarily (leading to a lower precision), catching as many at-risk patients as possible is a priority.

The results showed that the tuned KNN model performed better than the basic KNN, and the logistic regression model provided a solid baseline performance. While no model is perfect, each of these models did a good job of predicting the recurrence class based on the data available.

I feel moderately confident in the models I built. The performance metrics were promising, especially the recall rates, which are crucial for this type of problem. However, I recognize that in a real-world clinical setting, further validation would be necessary. Additional cross-validation, testing on external datasets, and possibly even more feature engineering could help to improve the models further. In summary, while the current models provide a strong starting point, continuous evaluation and refinement are essential before any real-world application.