COE Project 1
Aidan Liu
Professor: Joe Stubbs

What did you do to prepare the data?:
To prepare the data, I first loaded the dataset from the csv file (project1.csv). Using pandas, I then could examine the shape and size to understand the overall dimensions. Then using the .info() function (df.info()), I was able to inspect the different data types in the csv file. Some of the data types included: “class”, "age", "menopause", "tumor-size", "inv-nodes", "node-caps", "breast", "breast-quad", “irradiat”. Everything except the “deg-malig” was classified as a categorical type, ensuring that the “irradiat” column was also appropriately handled as a boolean category. This was done using the astype function (astype(“category”)) method. Then I checked for duplicate rows using the sum and duplicated functions which showed how many duplicates there were then just removing all of those rows with the drop_duplicate functions (df.drop_duplicates(inplace=True, ignore_index=True)). To handle missing values, I used the isnull and sum function to show the total number of null values and then applied a group-based filler using the groupby function (groupby('deg-malig')) and transform function (transform(lambda x: x.fillna(x.mode()[0]))). Next up, I ran the describe function to generate summaries of all the statistics and used seaborn functions such as histplot function and countplot function to visually analyze the variables such as “deg-malig”, "age", "menopause", "tumor-size" etc. which helped understand the patterns in the data and identify anomalies in the data. Finally using one-hot encoding, I changed the categorical variables using the get_dummies function to transform each categorical column into binary variables which makes it easier for the machine to use. This ensured that the dataset is clean and ready for training/evaluation.

What insights did you get from your data preparation?:
From the data, there were several key insights that I saw. Firstly, by examining the shape of the data with the shape function, I found that the dataset was small but had a lot of categorical variables. This made me convert those variables into the categorical type using the astype function. I also discovered the missing values in “tumor-size” and “inv-nodes” which made me fill in those null values. I did so by grouping “deg-malig” and imputing missing values with the groupby function and transform function. Those functions filled in the null values and reinforced that these features are tied into the tumor’s overall aggressiveness. Through the univariate graphs such as the histplot function and countplot function I saw that most of the patients are middle-aged, the tumor size was also moderate, and the lymph nodes were usually not involved. These observations provided an overall sense of the disease’s typical progression and the demographic it impacted. Finally I used one-hot encoding with the get_dummies

function to transform the categorical data to binary values, which makes it ready for machine learning. These insights validated the underlying patterns in patient demographics and characteristics of the disease that would make building an effective model better.

What procedure did you use to train the model?:
For training the models, I first split the dataset into training and test sets using the train_test_split function. I used a test size of 0.3 of the class and a training size of 0.7 of the class. I also used a fixed random_state so I could reproduce the outcomes. There were 3 different classification methods that I used. Firstly, I used the K-Nearest-Neighbors classifier. I initially trained the model with n-neighbors of 3 with the fit function. Secondly, I used the GridSearchCV classifier which optimizes the overall accuracy by searching a range of neighbor values from 1 to 99 and then performing a separate search on the grid to enhance the model's capability to catch positive cases. The best model of n-neighbors was ultimately selected on cross-validating the results. For linear classification, I trained the SGDClassifier with a perceptron loss and an alpha value of 0.01 with the fit method. Finally I printed out the accuracy, recall, precision, and f1 scores of all 3 models to compare their overall performances.

How does the model perform to predict the class?:
The model performs rather poorly sometimes and for other times its moderately average. For example, for the KNN classifier with k=3 the model had a test accuracy of 62%, recall of 0.25, and a precision of 0.36 for the positive cases, indicating that it does moderately on the negative cases and struggles to identify the positive cases. With the GridSearchCV it did slightly better in accuracy with 67% but the recall for positive cases was 0% which meant that it basically didn't identify any positive cases. This might have been due to the fact that the metric used for optimization is focused on accuracy over recall, leading to the low recall rate. It could also be because of the high K value which would smooth out the decision boundary and not capture any finer distinctions needed to identify the positive cases if they are rare. The linear classifier resulted in an overall accuracy of 57%, recall of 0.06, and precision of 0.12 for the positive cases. This suggests that the model wants to achieve moderate overall accuracy. Overall, the model performs poorly to identify the missing positive cases.

How confident are you in the model?:
Based on the evaluation metrics, I'm not that confident in the model. Although the accuracy may seem average, the low recall for positive cases is not great. This means that the model might fail to identify the positive cases, and in this case for medical diagnosis, the lack of identification of the positive cases may have serious consequences.