CSCI-GA 3033-061 Predictive Analytics
Analysis of Feature Selection and Classification for Breast Cancer

Lucas Maritnez
lom2017@nyu.edu

(a) Feature Engineering and Preprocessing

First, the categorical variables were encoded using `LabelEncoder` from scikit, mapping them to numerical values while ensuring that `NaN` and `None` values were not altered. This step is crucial since machine learning models work with numbers.

Then missing values were replaced using K-Nearest Neighbors (KNN) imputation with k=5. This approach ensures that missing data points were estimated based on the mean of the five nearest neighboring values. For (ex)categorical features, the value was rounded to the nearest integer ensuring it remains within the expected range.

After filling-in the missing values, I detected outliers using the Local Outlier Factor (LOF) method. Since only around 5% of the data were detected as outliers, I decided to simply remove them from the working dataset, keeping the remaining 95%.

Standardization was then applied to the numerical/continuous features, scaling them to a standard normal distribution to improve the performance of machine learning models.

Finally, during feature selection, the entropy was evaluated to rank features based on their importance. After that, I performed a Sequential Feature Selector, with a Logistic Regression model, to keep 70% of the (most important) attributes, i.e. the top 10 of the 15 attributes.

(b) Results from Models and Feature Selection/Ranking

Models and Accuracy: An 80/20 split was used to separate the training and test sets. The following models were trained and evaluated:

Classifier Model	**Accuracy**
K-Nearest Neighbors (k=5)	0.8863
Naive Bayes	0.8444
C4.5 Decision Tree	0.7373
Random Forest	0.9111
Gradient Boosting	0.9020
MLP Neural Network	0.9007

Table 1: Model accuracy comparison for different classifiers

Feature Selection, Ranking, and Importance: Feature selection was performed to evaluate the importance of various features using entropy and logistic regression for ranking, as described in **section (a)**. The top 10 features selected were:

1. Age
2. Race
3. Marital Status
4. N Stage
5. 6th Stage
6. differentiate
7. Grade
8. Estrogen Status
9. Regional Node Examined
10. Survival Months

Later, after optimizing hyperparameters as described in the next section, I explored the features' importance of the Gradient Boosting Classifier. The following image highlights the top features that contributed to classification decisions:

Rank	Feature	Importance
1	Survival Months	0.813418
2	Age	0.051888
3	6th Stage	0.041943
4	N Stage	0.032445
5	Estrogen Status	0.023732
6	differentiate	0.011798
7	Regional Node Examined	0.010068
8	Grade	0.008025
9	Race	0.006682
10	Marital Status	0.000000

Figure 1: Top features contributing to classification decisions in the optimized Gradient Boosting Classifier

The MLP Classifier does not inherently provide feature importance, so it was not analyzed.

(c) Results from Hyperparameter Search

Hyperparameter optimization was performed on the Gradient Boosting and MLP Classifiers, using the Random Search method with 1000 samples of parameter settings. I selected this optimization method, which, although potentially less precise due to its random nature, is significantly faster than the Grid Search method, which would be too time-consuming. The following hyperparameters were tuned:

Gradient Boosting Classifier

- `n_estimators`: 5 to 150 (steps of 5)
- `max_features`: 0.1 to 1.0 (steps of 0.01)
- `learning_rate`: 0.01 to 1.0 (steps of 0.05)

Where `n_stimators` is the number of boosting stages to perform, `max_features` is the fraction of features to consider when looking for the best split, and the `learning_rate` which is self-explanatory.

The optimization resulted in an accuracy of 0.0072, with the following hyperparameters values:

- `n_estimators`: 15
- `max_features`: 0.8399999999999996
- `learning_rate`: 0.26

MLP Classifier

- `learning_rate_init`: 0.001 to 1.0 (steps of 0.05)
- `alpha`: 0.0001 to 0.1 (steps of 0.005)
- `max_iter`: 1000 (constant)

Where `learning_rate_init` is the initial learning rate used, and `alpha` is the strength of the L2 regularization term. Also, `max_iter` (number of max iterations to run) was set to a constant value of 1000.

The optimization resulted in an accuracy of 0.0072, with the following hyperparameters values:

- `learning_rate_init`: 0.051000000000000004
- `alpha`: 0.0251
- `max_iter`: 1000 (constant)

Model Performance Summary

Gradient Boosting Classifier:

Metric	Score (Unoptimized)	Score (Optimized)
Accuracy	0.9020	0.9085
Precision	0.9265	0.9258
Recall	0.9618	0.9710
F1 Score	0.9438	0.9478

Table 2: Performance of the Gradient Boosting Classifier with and without optimized hyperparameters

MLP Classifier:

Metric	Score (Unoptimized)	Score (Optimized)
Accuracy	0.9007	0.9098
Precision	0.9226	0.9174
Recall	0.9649	0.9832
F1 Score	0.9433	0.9492

Table 3: Performance of the MLP Classifier with and without optimized hyperparameters

(d) Conclusions

1. **Model Performance:** The unoptimized Random Forest showed the best overall performance, achieving an accuracy of 91.11%. After hyperparameter tuning using Random Search Optimization, both the Gradient Boosting and MLP Classifiers reached accuracies close to that of the Random Forest (90.85% and 90.98%, respectively).
2. **Feature Importance:** Although the top 10 ranked features were selected for training, the Gradient Boosting Classifier relied heavily on the 'Survival Months' feature, which had an importance score of 0.81. This might suggests that applying PCA for further dimensionality reduction could be beneficial for this classifier, but additional investigation would be needed to confirm this hypothesis.
3. **Hyperparameter Tuning:** Optimization slightly improved the accuracy and F1 scores for both the Gradient Boosting and MLP Classifiers. This small change and relatively high accuracy suggests that the default hyperparameters were already close to optimal, although applying Grid Search could provide more insight.