Group #17: Micah Fadrigo (Student ID # 42923836), Cecilia Nguyen (Student ID # 44328584)
Dataset: Breast Cancer Wisconsin Diagnostic Data Set | Source: Data Source

Final Project Report

Introduction

Cancer is a disease in which abnormal cells in the human body grow uncontrollably in the absence of biological signals. Breast cancer cells can manifest as a lump of tissue in the breast, which leads to the development of malignant tumors, which are cancerous. Depending on the stage of cancer, malignant tumors may spread to other parts of the body, disrupt vital organ functions, and destroy healthy cells - decreasing the likelihood of survival. Benign tumors manifest as an overgrowth of normal cells, but are non-cancerous and non life-threatening. Factors such as a patient's diagnosis, the type of tumor they have, and how early the tumor is identified, will determine their treatment and their chance of survival. A more accurate and timely diagnosis on patient outcomes with machine learning is a matter of life-or-death and can revolutionize the healthcare industry. We will use features from the dataset above, which are computed from digital images of breast mass. There are multiple cell nuclei in each breast mass and each cell nucleus has 10 main characteristics (i.e. radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension). The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features (i.e. mean radius, standard error radius, worst radius).

EDA and Feature Preprocessing

The original dataset contained 569 data points and 33 features. One of the features contained all null values, and so we removed it. Patient ID was also an unnecessary feature that we did not use in any of our models. We changed diagnosis levels, benign and malignant, into 0 and 1, respectively. Since our dataset is high-dimensional, we performed feature selection to improve model performance, reduce computational complexity, and reduce noise. Structural multicollinearity was present due to the standard error and worst features being constructed from the mean features. As a result, similar information would be shared between the sets of features, potentially resulting in unreliable estimates for how the features would affect the response variable (diagnosis). Thus, a majority of our models start from a dataset that contains only the mean features. Another big concern was imbalanced data, where benign was the majority class (63%) and malignant was the minority class (37%).

In cases where models were not robust to differences in feature scales (such as SVM), we used a standard scalar to transform the features to follow standard normal distribution with mean = 0 and variance = 1. Standardization of features was also employed prior to using regularization techniques such as Lasso cross-validation. This allowed features to be comparable and helped prevent models and regularization techniques from being biased towards features with extreme magnitudes. If model performance on test data was not a high percentage (~90%), we experimented with adding features outside of the mean features.

1 SVM

SVM is a supervised algorithm that can be utilized for classification. SVM increases the dimensionality of data in order to find a hyperplane to separate the two classes. To increase dimensionality, a kernel function is used to assign new coordinates of the data. The regularization parameter C controls the softness of the margins and decision boundary broadness. The gamma parameter is for the rbf kernel, to compute the similarity between input data points.

1.1 Preprocessing Phase

Since SVM can be sensitive to differences in features scales, our mean features were standardized. We also included the regularization parameter values when creating our SVM classifier, C, in our model, to discourage the model from relying too heavily on any feature. Additionally, we utilized cross-validation methods to select the most appropriate kernel function to mitigate the effects of correlation.

1.2 Training & Hyperparameter Tuning

We began with first splitting the model into train and test sets. To deal with imbalanced data, the weighted class SVM was implemented in order to assign higher penalties to misclassified training instances of the minority (malignant) class. To find optimal parameters and perform hyperparameter tuning, two cross validation methods were utilized. For the first method, general cross-validation, a parameter grid/search space was defined with a range of values for each of the following hyperparameters: kernel, C, and gamma (shown in first table below; 2nd column). General cross-validation was employed to determine the best combination of model parameters from the search space as well as the required dimensions (shown in first table below; 3rd column).

Upon finding that the best kernel function generated from the first method was rbf, we tested a range of values for C and gamma while using the rbf kernel (using the second hyperparameter tuning method, Bayesian optimization). Bayesian optimization ran in a reasonable time compared to Grid Search and Random Search. The second table below shows the optimal values it generated for C and gamma (3rd column).

Name of Hyperparameter	Hyperparameter Values	Optimal Values Determined By General Cross Validation
Kernels	linear,poly,rbf	rbf
C	(.001,0.01,0.1,1,10,100)	1
gamma	range(0.001,0.01,0.1,1,10,100)	0.1

General Cross Validation

Name of Hyperparameter	Hyperparameter Values	Optimal Values Determined By Bayesian Optimization
C	(0.1,100)	1.78
gamma	(0.01,10)	0.01

Bayesian Optimization

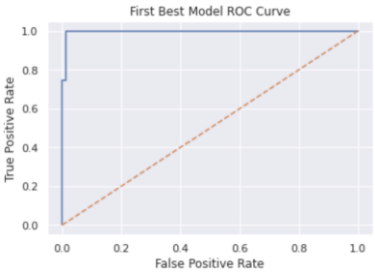
1.3 Experimental Results (Note: First See Table in Conclusion for Comparison of Model Performance on Test Data)

After conducting General Cross Validation and Bayesian optimization on the train data, it determined the optimal values for the hyperparameters (shown above). Our two models utilized the default cross validation parameter of 5 folds, which was then applied on the test set to measure the model's performance using the optimal hyperparameter choices.

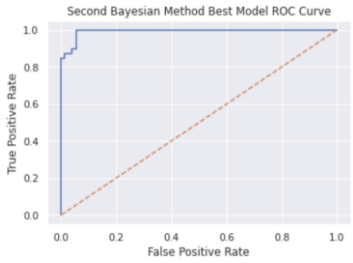
Referring to the performance metric (classification report) of the optimal hyperparameters for test data, for medical diagnosis, we prioritize high precision (ability to avoid false positives) and high recall (ability to avoid false negatives). F1 score is a model performance measure that captures both, in which an F1 score of 1 indicates perfect precision and recall. The classification reports are given above.

For General Cross Validation SVM, the model performed well on unseen data, with high precision and high recall, as indicated by the F1 score of 0.91 for malignant and 98% testing accuracy. For Bayesian Optimization, the model also performed well on unseen data, with high precision and high recall, as indicated by the F1 score of 0.92 for malignant and 98% testing accuracy. However, for the General Cross Validation SVM, a high recall score of 1.00 would mean that the model had correctly identified all patients who have the disease, but it can also classify healthy patients as having a tumor, leading to false positives that can lead to unnecessary/harmful treatment for healthy patients.

To check for possible overfitting we can compare the F1 scores of training and test data. If both F1 scores are high, it means that the model is probably not overfitting and able to generalize well to new data. For both models, they both have high F1 values for train/test sets, meaning that the classifiers can generalize well to new data.



AUC: 0.997



AUC: 0.993

Another alternative to visualize the performance of our classification model would be to look at an ROC/AUC curve, where our y-axis is the True Positive Rate (recall) and our x-axis is the False Positive Rate (1-precision). Since we have an imbalance in the classification of tumors, we would like to utilize ROC/AUC since it is a more robust evaluation, and provides a more comprehensive measure. The diagonal line represents a random classifier which makes random guesses of the classes in a 50-50 manner. Our ROC curve is around 1, indicating a high classification ability. Looking at the AUC's (Area under the ROC curve), it is also high (~99%). Both of these measurements indicate that our classifier model has a high classification ability to separate benign/malignant tumors.

2 Random Forest

Random Forest is an ensemble learning algorithm that combines multiple decision trees, each trained on a random subset of features and the data. The final prediction is made by aggregating the predictions of all individual trees. These characteristics allow random forest to be able to prevent overfitting, generalize well to unseen data, and reduce the effect of correlated features.

2.1 Preprocessing Phase

Since random forest reduces the effect of correlated features to some extent, all of the "mean" type features were used and highly correlated features were not removed.

2.2 Training & Hyperparameter Tuning

Imbalanced data can reduce the effectiveness of random forest by leading to biased selection of features that are more prevalent in the majority class, poor tuning of hyperparameters, and misclassification of the minority class. To address class imbalance, we used Stratified K-fold cross validation, and GridSearchCV combined. Stratified K-fold cross validation is essentially K-fold cross validation, except that it ensures that within each fold, the proportion of each class is roughly equal. We also used GridSearchCV for hyperparameter tuning which will search over a range of hyperparameters to find the combination that yields the best performance on the validation set. Using these two methods together will allow us to make accurate hyperparameter choices.

The data was split into train and test sets and a parameter grid was defined, which consisted of different values for the following hyperparameters: number of decision trees, the maximum number of features randomly selected for each base learner, the maximum depth of each base learner, the minimum number of samples required to be in a leaf node, and the criterion to evaluate the quality of a split. After conducting GridSearchCV on the training set, it determined the optimal values for the hyperparameters (shown in first table). Stratified 10 fold cross validation was then applied on the training set to score the selected combination of hyperparameters (second table).

Name of Hyperparameter	Hyperparameter Values	Optimal Values Determined By Grid Search CV
n_estimators	[50, 100, 200]	200
max_features	range(1, 11)	4
max_depth	range(2, 15)	7
min_samples_leaf	range(1, 3)	1
criterion	[‘gini’], [‘entropy’]	‘gini’

Precision	0.913
Recall	0.944
F1 Score	0.925
Accuracy	0.941

Scores of Best Combination of Hyperparameter Values (Grid Search) for Random Forest

2.3 Experimental Results (Note: First See Table in Conclusion for Comparison of Model Performance on Test Data)

The random forest model hyperparameter choices generated from GridSearchCV is expected to perform well, after using 10-fold cross-validation, with high precision and high recall, as indicated by the F1 score of 0.93 and 96% accuracy (seen above). Its performance measures on the test data are all above 90% with a F1 score of 0.94 and 96% accuracy. See conclusion for more details on test performance.

3 Gradient Boosting (AdaBoost)

AdaBoost is a gradient boosting algorithm that, similar to random forest, averages over many weak base learners to produce a strong classifier. AdaBoost works by iteratively training weak learners on different subsets of the training data and adjusting the weights of the training examples to emphasize the examples that are misclassified by the current set of weak learners. AdaBoost is suitable for our problem because it handles imbalanced data (to some degree) by assigning higher weights to misclassified samples (the minority class) in order to improve overall accuracy. However, a drawback is that when the dataset contains highly correlated features, the weak learners in AdaBoost may place too much emphasis on those features which contribute similar information to the model. This can lead to overfitting and reduced generalization ability.

3.1 Preprocessing Phase

For the dataset belonging to the **AdaBoost classifier 1**, highly correlated features were removed from the “mean” type features. Six features in total were used for the first **AdaBoost classifier 1**. In an attempt to improve this model, these features were later scaled prior to conducting Lasso cross validation, however, no features were eliminated. Six features in total were used for the **AdaBoost classifier 1**: `radius_mean`, `compactness_mean`, `texture_mean`, `smoothness_mean`, `fractal_dimension_mean`, and `symmetry_mean` (ranked from most to least feature importance according to Lasso CV). For the dataset belonging to the **AdaBoost classifier 2**, highly correlated features were removed from all type features (“mean”, “standard error”, and “worst”). The features for **AdaBoost classifier 1** are a subset of the features for **AdaBoost classifier 2**. In an attempt to improve this model, these features were later scaled prior to conducting Lasso cross validation, however, no features were eliminated. Ten features in total were used for the **AdaBoost classifier 2**: `radius_mean`, `symmetry_worst`, `texture_mean`, `compactness_mean`, `smoothness_mean`, `symmetry_se`, `fractal_dimension_mean`, `texture_se`, `symmetry_mean` and `smoothness_se` (ranked from most to least feature importance according to Lasso CV).

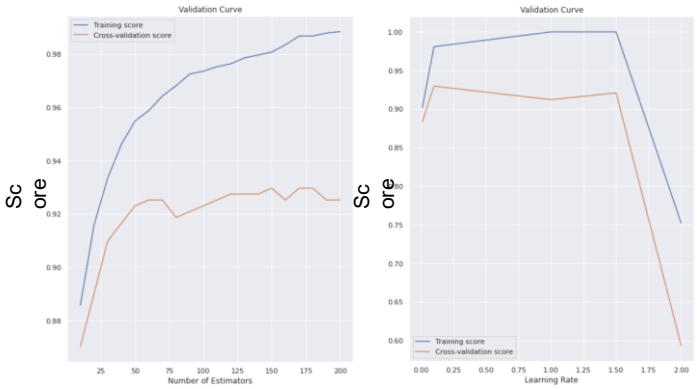
3.2 Training & Hyperparameter Tuning

The number of base learners (`n_estimators`) is the number of decision stumps and the learning rate (`learning_rate`) is the factor that scales the contribution of each base learner before adding it to the final ensemble. A small learning rate means that the classifier will take smaller steps towards the optimal solution and may require more iterations to converge, however it may result in high accuracy and low variance. A large learning rate means that the classifier will take bigger steps towards the optimal solution and may require fewer iterations to converge, however it may result in low accuracy and high variance. It is necessary to tune both hyperparameters to balance the bias-variance tradeoff.

Below are **two sets** of plots for **AdaBoost classifier 1** and **AdaBoost classifier 2** (respectively)**,** where each plot shows the average training and validation scores over multiple iterations for each value of the hyperparameter. For **AdaBoost classifier 1,** when **`n_estimators` = 190** and **`learning_rate` = 0.1**, model performance is maximized without overfitting the training data. GridSearchCV also verified that these are the optimal values. For **AdaBoost classifier 2**, when **`n_estimators` = 150** and **`learning_rate` = 0.1**, model performance is maximized without overfitting the training data. GridSearchCV also verified that these are the optimal values.

Precision	0.935
Recall	0.918
F1 Score	0.923
Accuracy	0.930

Scores of Best Combination of Hyperparameter Values (Grid Search) for AdaBoost Classifier 1



Precision	0.913
Recall	0.944
F1 Score	0.925
Accuracy	0.941

Scores of Best Combination of Hyperparameter Values (Grid Search) for Adaboost Classifier 2

3.3 Experimental Results (Note: First See Table in Conclusion for Comparison of Model Performance on Test Data)

Because the hyperparameter choices of **AdaBoost classifier 2** outperformed those of **AdaBoost classifier 1** on all performance measures, we wanted to improve upon **AdaBoost classifier 2**, specifically by making its base learners more complex. Keeping `n_estimators` = 80 and `learning_rate` = 1.5, we tested `max_depth` values ranging from one to five using GridSearchCV. It determined the optimal value was four. Although performance on the test data did not improve by increasing maximum depth to four, this **variation of AdaBoost classifier 2** still outperformed **AdaBoost classifier 1** on all performance measures. Similar patterns were detected when testing all of their performance on the test data. See conclusion for more details on test performance.

Conclusion

Model	Precision	Recall	F1 Score	Accuracy
SVM Classifier 1 (General)	0.95	1.00	.94	.98
SVM Classifier 2 (Bayesian)	0.97	0.87	.95	.98
Random Forest	0.947	0.923	0.935	0.956
AdaBoost Classifier 1 (n_estimators = 190, lr = 0.1, maxDepth = 1)	0.927	0.809	0.864	0.895
AdaBoost Classifier 2 (n_estimators = 150, lr = 0.1 maxDepth = 1)	0.953	0.872	0.911	0.930
AdaBoost Classifier 2 (n_estimators = 80, lr = 1.5, maxDepth = 4)	0.951	0.830	0.886	0.912

Model Performance on Test Data

SVM, Random Forest, and AdaBoost are all popular machine learning algorithms that can be used for medical diagnosis classification tasks. Random Forest was considered for this problem due to its robustness to highly-correlated features and to overfitting. However, it's not ideal for imbalanced data, and so Stratified K-fold had to be employed before finding the optimal parameters. On the contrary, AdaBoost handles class imbalance really well (given there are no influential outliers) by reweighting misclassified samples, but its effectiveness is limited in the presence of highly correlated features. SVM was considered because it's ideal for when the dataset is small (only 569 observations), the number of features is relatively low (considering only 10 mean features), and there is a need for high classification accuracy (important in medical diagnosis). SVM Classifier 1 was the best model with the highest recall and accuracy. 100% of the patients that had malignant tumors were correctly classified as positive, and about 98% of patients were classified correctly. It had the second-highest precision score, in which 95% of the patients who were classified as positive, actually had a malignant tumor (5% false positives). SVM Classifier 2 performed similarly on all performance measures except recall, in which only 87% of the patients that had malignant tumors were correctly classified as positive (13% false negatives). Random Forest also performed similarly to SVM Classifier 2 with performance measures all greater than 90%. The AdaBoost classifiers had similar precision performance to the random forest model, however, had lower performance in recall, F1-score, and accuracy. Of all the AdaBoost classifiers, AdaBoost Classifier 2 had the best performance on all measures. With 4 more additional features and 40 less base learners compared to AdaBoost Classifier 1, it performed better given a small learning rate of 0.1. Increasing the complexity of each base learner for AdaBoost Classifier 2, however, resulted in lower values for recall and F1-score that were both less than 0.90.

Team Contributions

Micah Fadrigo and Cecilia Nguyen conducted initial data preprocessing and EDA, respectively. We listed our main concerns with the dataset: class imbalance, highly correlated features, and features with different scales, and determined potential classifiers and methods combined that would address those issues. Cecilia trained models using Support Vector Machines. Micah trained models using Random Forest and AdaBoost. We each explored different cross-validation methods for hyperparameter tuning, however, we shared the same test performance metrics to ensure proper comparison of models. In the end, we compared models based on how we adjusted them to fit our problem and dataset to the best of our ability, and finally compared performance measures.