Group #17: Micah Fadrigo (Student ID #42923836), Cecilia Nguyen (Student ID #44328584)

Final Project Report

Dataset: Breast Cancer Wisconsin Diagnostic Data Set
Source: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

Introduction

Cancer is a disease in which cells in the human body grow abnormally as the damaged cells grow and multiply in the absence of biological signals. These cells may disrupt vital organ functions through tumors and destroy healthy cells, resulting in death. Cancer is the second-most common death, after heart attacks. In the US, there are high annual incidence rates of breast cancer. Specifically, breast cancer has affected many women, being the second most common cause of death due to cancer after lung cancer.

Breast cancer cells may form tumors, lumps of tissue which can be classified as either malignant or benign, affecting a patients' diagnosis and further treatment. A more accurate and timely diagnosis on patient outcomes with machine learning can revolutionize the healthcare industry. Utilizing the Wisconsin Breast Cancer Diagnosis Data Set, we will be classifying malignant and benign tumors. The feature measurements were computed from digital images of breast mass, where the image describes the characteristics of the cell nuclei, such as the mean radius, concavity, perimeter, texture.

EDA and Feature Preprocessing

The dataset initially contains 18,208 data points, with 33 features. We will be transforming our dataset to be machine learning ready. Since one of the features is all null values, we exclude it from the dataset. We will also be label encoding the diagnosis features into 0/1. Since our dataset is large, we will also be performing feature selection to improve model performance, reduce computational complexity, and reduce noise.

Noticing mean features as having high correlations with other features, we selected all the mean features from the dataset. Since mean features are already built for standard error features and worst features. Similar information with standard error and worst features would cause multicollinearity and increase computational complexity. The mean features will also be part of the standardization of the variables to allow for our model performance to not be biased due to extreme magnitudes between features. Standardization using a standard scalar would transform the features to follow standard normal distribution with zero mean and variance 1, and therefore make the features be comparable. If necessary, if model performance on test data is not a high percentage (~90%), we will add features most important.

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 through finding a hyperplane. 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 highly correlated data, our mean data features were standardized to decorrelate the features. 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.

1.1 Training and classification

Imbalanced data also had to be dealt with, where 63% of observations belong to the benign class and 37% belong to the malignant class, the weighted class SVM was implemented in order to assign higher misclassification penalties to training instances for our smaller malignant class.

To find optimal parameters and perform hyperparameter tuning, we utilized two cross validation methods. We began with first splitting the model into train and test sets. We then defined a parameter grid with the following hyperparameters. In our first method included we defined a search space with a range of kernel, gamma, and C values. The parameters will give us the best model given our parameters chosen, and this cross validation will help in also determining the dimensions required.

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

Finding that the best model from the first method included an rbf kernel function, we used the parameter in our model and implemented Bayesian optimization for cross validation, which runs in a reasonable time compared to the other methods with long runtime, such as Grid Search and Random Search.

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

1.1 Experimental results (Note: See Table in Conclusion for Comparison of Models)

After conducting General Cross Validation and Bayesian optimization on the train data, it determined the optimal values for the hyperparameters (shown below). 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.

Precision	Recall	F1 Score	Accuracy
0.95	1.00	0.94	0.98

Scores of Best Combination of Hyperparameter Values (General Cross Validation)

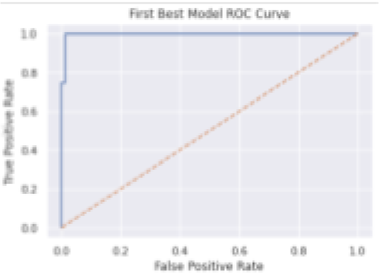
Precision	Recall	F1 Score	Accuracy
0.97	0.87	0.95	0.98

Scores of Best Combination of Hyperparameter Values (Bayesian Optimization)

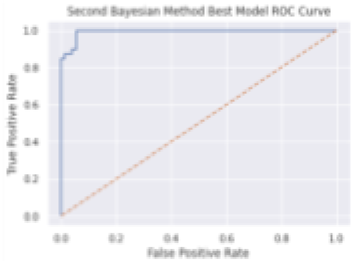
Referring to the performance metric (classification report) above for test data, for medical diagnosis, we prioritize high precision (ability to avoid false positives) and high recall (ability to avoid false negatives). F1 score is a model performance measure that captures both, in which an F1 score 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 100 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. We would be looking at the classification reports for train and test. 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.

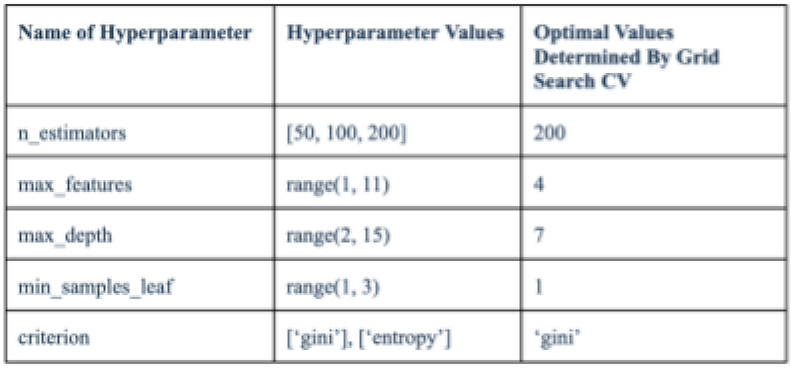
2.1 Preprocessing Phase

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.

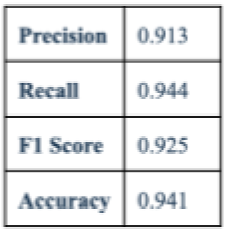
2.2 Training & Hyperparameter Tuning

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: See Table in Conclusion for Comparison of Models)

In the context of medical diagnosis, it is important to 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 random forest model with hyperparameter choices generated from GridSearchCV is expected to perform well on unseen data, with high precision and high recall, as indicated by the F1 score of 0.93 and 96% accuracy (seen above).

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

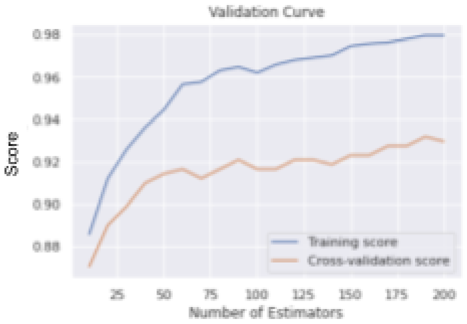
For the dataset belonging to the **AdaBoost classifier 1**, highly correlated features were removed from the “mean” type features in which perimeter_mean, area_mean, concavity_mean, and concave_points_mean were removed. Six features in total were used for the first **AdaBoost classifier 1**. In an attempt 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).

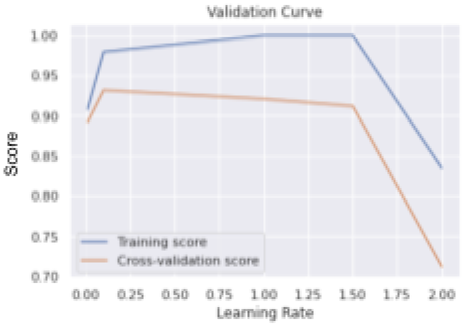
For the dataset belonging to the **AdaBoost classifier 2**, highly correlated features were removed from all type features (“mean”, “standard error”, and “worst”). The features for **AdaBoost classifier 1** are a subset of the features for **AdaBoost classifier 2**. In an attempt to improve this model, 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).

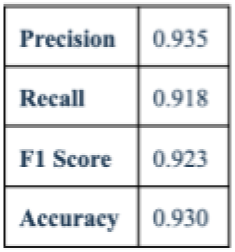
3.2 Training & Hyperparameter Tuning

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 plots for **AdaBoost classifier 1** that show the validation curve which shows the average training and validation scores over multiple iterations for each value of the hyperparameter. 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.



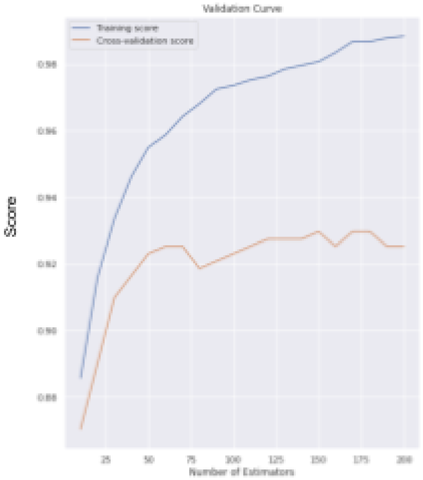


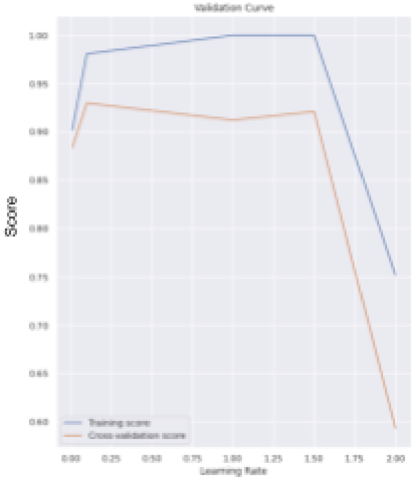


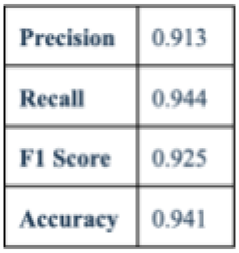
Precision	0.935
Recall	0.918
F1 Score	0.923
Accuracy	0.930

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

Below are similar plots 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	0.913
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: See Table in Conclusion for Comparison of Models)

Because **AdaBoost classifier 2** outperformed **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.

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

Performance of Classifiers on Test Data

Team Contributions

Micah and Cecilia 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 utilized Support Vector Machines to train her model. 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.