Breast Cancer Prediction using Breast Cancer Wisconsin Dataset

1st Ka Ho Chan
dept. of Computer Science
Stevens Institute of Technology
New Jersey, United States of America
kchan7@stevens.edu

2nd John Wright
dept. of Computer Science
Stevens Institute of Technology
New Jersey, United States of America
jwright8@stevens.edu

***Abstract*—This project aims to improve the early diagnosis of breast cancer by using machine learning algorithms to classify tumors as malignant or benign using the Breast Cancer Wisconsin (Diagnostic) dataset [1]. The dataset contains 569 samples with 30 features each, which describe physical characteristics of the tumors. We implemented several machine learning models including Logistic Regression, Decision Tree, Support Vector Machine (SVM), Bagging Logistic Regression, XGBoost, AdaBoost SVM, and a Stacked Ensemble. The Stacked Ensemble model, which combines the predictions of all the other models, achieved the highest accuracy of 99.1%, outperforming individual models. This work demonstrates the effectiveness of using ensemble techniques, particularly stacking, in enhancing predictive performance.**

I. Introduction

Breast cancer is one of the most common and deadly forms of cancer for women. Early diagnosis can greatly improve patient outcomes [3]. This makes it a critical subject of research. There is a need for automated and reliable solutions to aid in the early diagnosis of breast cancer. This project addresses this need by exploring the potential of machine learning algorithms to accurately classify breast cancer tumors as malignant or benign using the Breast Cancer Wisconsin (Diagnostic) dataset, [1].

The Breast Cancer Wisconsin (Diagnostic) dataset, contains 569 samples, each with 30 features that describe various characteristics of cell nuclei, such as radius, texture, perimeter, area, and smoothness. The target variable indicates whether a tumor is malignant (M) or benign (B). The dataset is slightly imbalanced, with 357 benign samples and 212 malignant samples.

In this project, we implemented several machine learning algorithms for this binary classification problem. These algorithms include Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), Bagging Logistic Regression (BLR), XGBoost (XGB), AdaBoost SVM (ASVM), and a Stacked Ensemble (SE). Each of these models brings its own strengths to the table: LR is known for its interpretability and effectiveness in binary classification tasks, SVM excels in handling high-dimensional spaces, and ensemble methods like Bagging, Boosting, and Stacking combine the strengths of multiple models to improve overall performance. The SE, in particular, was designed to combine the predictions of all the other models to produce a final, more accurate prediction.

Extensive experiments and analyses were conducted to evaluate the performance of these models. The SE achieved the highest accuracy score of 99.1%, demonstrating its effectiveness in predicting breast cancer. While other models also performed well, the SE proved superior by combining the strengths of individual models to make a better prediction. The high accuracy achieved by our models is comparable to (in some cases better than) those reported in recent studies. This validates the effectiveness of our approach.

II. Related Work

Existing solutions can be broadly categorized into traditional machine learning models, ensemble methods, and deep learning.

Traditional Machine Learning Models:

The study by Hridoy et al. (2024) [5] evaluated the performance of several traditional machine learning models for breast cancer detection using the Wisconsin Breast Cancer (Diagnostic) dataset. The models included LR, SVM, K-Nearest Neighbors (KNN), Naive Bayes (NB), and DT. The study explored the use of hyperparameter optimization by using Grid Search and Random Search. KNN and LR achieved the highest accuracy, 99.42%. However, KNN was determined to be the most effective because it achieved a recall of 100%. The study highlights the importance of optimizing model parameters to enhance the accuracy and reliability of the models.

A strength of KNN is its simplicity. It does not require an explicit training phase, making it adaptable to a variety of problems. KNN's major drawback is its computational inefficiency, because it requires calculating the distance between each input and every instance in the dataset for each prediction. Additionally, KNN is sensitive to noisy data and irrelevant features, which can significantly degrade its performance. This makes feature selection and data preprocessing critical steps.

SVMs can model very high-dimensional spaces. They excel at creating complex decision boundaries. SVMs are robust to overfitting due to their regularization capabilities, which controls the complexity of the decision boundary. However, the computational resources required to train SVMs can be a limitation when applied to large datasets. Additionally, SVMs often need careful tuning of hyperparameters to achieve optimal performance. This can be time consuming.

LR is simple and interpretable. It provides probabilities for class predictions, making it useful in binary classification

tasks like breast cancer detection. LR performs well when the relationship between the features and the outcome is linear. However, it struggles with non-linear relationships.

One of the major advantages of NB is its ability to easily handle large datasets. It is also very effective in high-dimensional spaces. NB also has fast training and prediction times. However, a significant drawback of NB is its reliance on the assumption that features are independent of each other. This assumption can lead to lower performance when there are strong dependencies between features, especially if the dataset is small.

DTs are intuitive and easy to visualize. This makes them useful for understanding the decision making process. They handle both numerical and categorical data well and can capture non-linear relationships. However, DTs are prone to overfitting unless properly pruned or regularized. They also tend to have lower accuracy than more sophisticated models.

Ensemble Models:

The study by Hridoy et al. (2024) [5] implemented Random Forest (RF) and XGB models, and both achieved an accuracy of 97.08%. The study noted that XGB was the only model that did not improve from the use of grid search. Eventhough the traditional models performed the best in this study, the authors could still have used the best performing individual models as the base models for a Stacked Ensemble. This would likely further improve the performance.

XGB is known for its superior performance and speed. One of the strengths of XGB is its ability to handle both linear and non-linear relationships in data. XGB is particularly well-suited for large datasets, because it is optimized for parallel processing. Additionally, XGB's ability to weigh the importance of each feature in the dataset is another advantage. However, XGB can be computationally intensive when dealing with very large datasets or when extensive hyperparameter tuning is required. XGB is also less interpretable than simpler models, because it can be difficult to understand the contribution of individual features to the final prediction.

One of the advantages of RFs is the ability to reduce overfitting, which is a common issue with individual decision trees. By averaging the results of many trees, RFs tends to produce more generalizable models. Additionally, RFs can handle both numerical and categorical data effectively. They are also effective for datasets with a large number of features. RFs provides feature importance metrics. This helps identify which features contribute most to the model's predictions, making it useful for feature selection. However RFs can be computationally intensive when a large number of trees are used. Also, RFs interpretability is lower than that of simpler models.

The study by Kumar et al. [2] implemented a SE that achieved an accuracy of 99.45%. The base models included: AdaBoostM1, gradient boosting, stochastic gradient boosting, CatBoost, and XGB. One of the advantages of SE is its ability to combine the strengths of various algorithms, allowing it to perform better than any single model. By combining models that have different strengths and weaknesses SEs can create a more accurate and robust model. One disadvantage is the increased complexity. It can also be computationally expensive to train multiple models and then combine them. Additionally, it is difficult to understand the contribution of each individual model to the final prediction.

Deep Learning Models:

The study by Zhang (2024) [4] discusses the application of a deep learning model with three hidden layers to the Wisconsin Breast Cancer dataset. The article notes that both the deep learning model and the RF achieved a prediction rate of 97.4468% in accuracy.

Models such as this are often called Fully Connected Neural Networks (FCNNs). FCNNs have the advantage of being able to learn complex patterns in data without requiring extensive feature engineering. They are particularly powerful when dealing with large datasets and unstructured data such as images or text. However, for tabular data their performance might not surpass models like RFs. Disadvantages of FCNNs include their need for large amounts of data, significant computational resources, and a tendency to overfit if not properly regularized. Additionally, deep learning models can be seen as "black boxes," making them less interpretable compared to traditional machine learning models.

The study by Hridoy et al. (2024) [5] implemented a Multilayer Perceptron model (MLP) with grid search that achieved an accuracy of 98.25%. MLPs excel at modeling non-linear relationships in data, due to their deep architecture. However, training an MLP typically requires a large amount of data and significant computational resources, especially as the number of layers and neurons increases. Additionally, MLPs are prone to overfitting, particularly if the network is too large for the available data.

III. Our Solution

This section elaborates your solution to the problem.

A. Description of Dataset

The Breast Cancer Wisconsin (Diagnostic) dataset, obtained from the UCI Machine Learning Repository [1], consists of 569 samples with 30 features each. These features describe various characteristics of the cell nuclei present in digitized images of Fine Needle Aspirates (FNA) of breast masses. The target variable indicates whether a tumor is malignant or benign. The dataset contains 357 benign samples and 212 malignant samples, making it slightly imbalanced.

Features: The 30 features include the mean, standard error, and worst (largest) values of measurements of each of the following:

- Radius
- Texture
- Perimeter
- Area
- Smoothness
- Compactness
- Concavity

- Concave points
- Symmetry
- Fractal dimension

Data Preprocessing:

1) Dropping Non-Informative Columns:
   The dataset initially included an 'ID' column, which was dropped because it does not provide any predictive value for the classification task.
2) Mapping Diagnosis Labels:
   For the LR and SVM models, the target variable 'Diagnosis' was mapped to binary values, with 1 representing malignant and 0 representing benign. However, this was not done for the DT, because it is unnecessary.
3) Handling Missing Values:
   The dataset was checked for missing values, and none were found.
4) Standardization:
   To ensure that the features are on a similar scale, all feature values were standardized using scikit-learn's StandardScaler. This involved subtracting the mean and dividing by the standard deviation for each feature, resulting in a dataset where each feature has a mean of 0 and a standard deviation of 1. Standardization is necessary for algorithms like LR, SVM, and others that are sensitive to the scale of the data. The DT model used the unstandardized data, because this model is not sensitive to the scale of the data.
5) Train-Test Split:
   The dataset was split into training and testing sets, with 80% of the data used for training and 20% for testing. The split was stratified to ensure that the class distribution was maintained in both the training and testing sets, and the data was shuffled to avoid any ordering bias.

A correlation matrix was generated to examine the relationships between the features, shown in Fig. 1. The correlation matrix shows a high amount of correlation between several features in the dataset. Features related to the size of the cell nuclei are strongly correlated with each other. Features related to shape also have a high amount of correlation. This could lead to multicollinearity.

Principal Component Analysis (PCA) was tested as a means to control potential multicollinearity by reducing the dimensionality of the data. Various values were tested for the n_components parameter. Low values resulted in a decrease in the accuracy of the models, and higher values resulted in no change to the accuracy. Therefore, PCA was not used. This is likely due to L2 regularization sufficiently handling any potential multicollinearity.

To gain insights into the data, we first displayed the first 5 rows of the data. Then, we generated box plots for each feature. The box plots revealed that there was a large amount of variance in the scale of different features. This was handled by scaling the data as mentioned above. Outliers were observed in the features. This may decrease the performance of the models. Additionally, skewness was observed in some

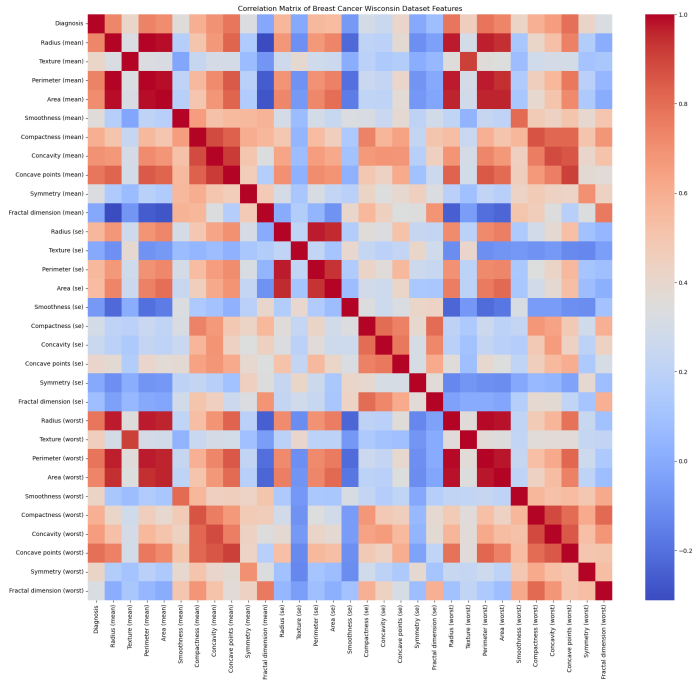

Fig. 1. Correlation matrix of the features.

features, suggesting that transformations might be beneficial to normalize the data and improve model performance.

However, due to time constraints and the already complex nature of this project, we have decided to leave outlier detection and the transformation of skewed features for future work.

The dataset is slightly imbalanced, with a higher number of benign samples than malignant ones. Therefore, stratified sampling was used during both the train-test split and the cross-validation done during grid search (we used scikit-learn's GridSearchCV and this uses stratified sampling by default for models that are classifiers). This ensures that each fold used in the evaluation of the model maintains a balance of the classes. This prevents bias towards the majority class.

B. Machine Learning Algorithms

LR: This is a linear model designed for binary classification tasks. It calculates the probability of a class by applying the logistic function to the features. The model's coefficients represent the significance of each feature in the prediction. This algorithm is suitable for our problem, because it helps to understand the impact of different features on the diagnosis of breast cancer and it is effective for binary classification problems. Initially, we used the following hyperparameters: 'penalty': 'l2', 'C': 0.1, 'solver': 'liblinear', 'max_iter': 100.

DT: These split the data into subsets based on the value of a selected feature. Each node in the tree represents a feature, and each branch represents a decision rule. The leaves (nodes with no further branching) represent the prediction. Decision Trees can handle both numerical and categorical data and

can capture non-linear relationships between features and the target variable. They are highly interpretable due to the visual representation of decision rules, and they are robust to outliers. For all of these reasons, DTs are suitable for this problem. Initially, We used the Gini impurity as the criterion for splitting nodes and set the maximum depth = 10 to avoid overfitting.

SVM: These are classification models that find the optimal hyperplane that separates data points of different classes in (potentially) very high-dimensional spaces. With the use of kernel functions, SVMs can model non-linear decision boundaries. SVMs are also robust against overfitting. For these reasons, SVMs are a great choice for this problem. Initially, We used a value of 0.1 for the C parameter to control the trade-off between a low error on the training data and minimizing model complexity (bias-variance tradeoff). We also used a value of 0.1 for the gamma parameter for the RBF kernel to define the influence of a single training example.

BLR: Bagging (Bootstrap Aggregating) is an ensemble method that combines the predictions from multiple models to improve reliability and accuracy. For this approach, we used the LR model as the base estimator. Using bagging with LR helps to avoid overfitting, by training multiple Logistic Regression models on different random subsets of the training data and averaging their predictions. For these reasons, this model is suitable for this problem. Initially, the values of the hyperparameters were: 'bootstrap': True, 'max_features': 1.0, 'max_samples': 0.1, 'n_estimators': 10.

XGB: This model is an ensemble method that builds models sequentially. Each model corrects the errors of its predecessor. XGB uses Decision Trees as the base model, which allows it to capture complex relationships in the data. Initially, We used the max_depth = 10 parameter to control the complexity of the model. We also used the learning_rate = 0.1 and the n_estimators = 100 parameters to control the step size at each iteration and the number of boosting rounds, respectively. This model's ability to handle imbalanced data (like this data) is another reason it is suitable for this problem.

ASVM: AdaBoost (Adaptive Boosting) is another ensemble method that combines weak classifiers into a strong classifier. For our problem, we used SVM as the base estimator for AdaBoost. This model focuses on the instances that are harder to classify, by assigning more weight to misclassified examples in each round. This can lead to better overall accuracy. Initially, the parameters we set were the learning_rate = 0.1, which controls the contribution of each classifier, and the n_estimators = 50, which defines the number of weak classifiers to combine.

SE: This model combines multiple models to leverage their individual strengths. We stacked all of the other models in this project. The predictions from each model were used to train the final LR model, which made the final classification decision. This approach is particularly effective because it allows us to capture a diverse set of patterns from the data by combining the strengths of the different models, improving overall performance. Initially, we chose the same parameters that we did for the regular LR model.

Each of these algorithms was selected for certain aspects of our problem. LR and BLR were selected to capture linear relationships. SVM, XGB, and ASVM were selected to capture non-linear relationships. Then, combining these approaches into a SE should create a robust and accurate model.

C. Implementation Details

LR: This model was initialized without specific parameters, using the default settings using Scikit-learn's LogisticRegression class. Next, a grid search was performed (using Scikit-learn's GridSearchCV) over the values in the parameter grid shown in Fig. 2. This grid search was performed using 5-fold cross-validation, and the model with the best mean cross-validation score was selected. This ensures that the model selected with the best parameters generalizes well to unseen data. The verbose parameter was used with the grid search to display all of the intermediary results.

The parameters tuned included: C (inverse of regularization strength), max_iter (maximum number of iterations taken for the solvers to converge), penalty (type of regularization applied), and solver (algorithm to use in the optimization problem).

```
    'penalty': ['l1', 'l2'],
    'C': [0.01, 0.09, 0.1, 0.11, 1, 10],
    'solver': ['liblinear'],
    'max_iter': [100]
},
{
    'penalty': ['l1', 'l2'],
    'C': [0.01, 0.1, 1, 10],
    'solver': ['saga'],
    'max_iter': [100000]
},
{
    'penalty': ['l2'],
    'C': [0.01, 0.1, 1, 10],
    'solver': ['lbfgs', 'sag'],
    'max_iter': [6000]
```

Fig. 2. Parameter grid for Logistic Regression.

The best hyperparameters found for the LR model were: 'C': 0.09, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'. Then, the best model found in the grid search was retrained on the full training dataset and then evaluated on the test set. Model evaluation metrics for all the models include: mean CV score, standard deviation of the CV score, accuracy score, percision, recall, and f1-score. These metrics for each model are discussed in the Comparison section.

Additionally, the Logistic Regression coefficients are shown in Fig. 3. These show the importance of each feature in the classification of of the masses as malignant or benign.

DT: This model was initialized using Scikit-learn's DecisionTreeClassifier class with default settings. Next, a grid search was performed over the values in the parameter grid shown in Fig. 4. The grid search for this model and all of the other models is the same as explained above in the LR implementation. The only thing that differs are the values tested (the parameter grids). The parameters tuned included: criterion

	Coefficient
Texture (worst)	0.562815
Radius (worst)	0.489417
Area (worst)	0.476942
Radius (se)	0.463429
Perimeter (worst)	0.450167
Concave points (mean)	0.437579
Symmetry (worst)	0.427518
Concave points (worst)	0.406281
Texture (mean)	0.404190
Area (se)	0.390135
Area (mean)	0.372183
Radius (mean)	0.357274
Perimeter (mean)	0.349343
Perimeter (se)	0.346666
Concavity (worst)	0.344370
Smoothness (worst)	0.344327
Concavity (mean)	0.341745
Smoothness (mean)	0.177736
Concave points (se)	0.109731
Compactness (worst)	0.106793
Fractal dimension (worst)	0.094376
Smoothness (se)	0.058532
Symmetry (mean)	0.043016
Compactness (mean)	0.020939
Texture (se)	-0.042057
Concavity (se)	-0.064914
Symmetry (se)	-0.082386
Fractal dimension (mean)	-0.157930
Fractal dimension (se)	-0.226872
Compactness (se)	-0.252891

Fig. 3. Logistic Regression Coefficients.

(function to measure the quality of a split), max_depth (maximum depth of the tree), max_features (number of features to consider when looking for the best split), max_leaf_nodes (maximum number of leaf nodes), min_samples_leaf (minimum number of samples required to be at a leaf node), and min_samples_split (minimum number of samples required to split an internal node).

```
'criterion': ['gini', 'entropy'],
'max_depth': [10, 20, 30],
'min_samples_split': [2, 5, 10, 15],
'min_samples_leaf': [1, 2, 4],
'max_features': [None, 'sqrt', 'log2'],
'max_leaf_nodes': [5, 10, 20]
```

Fig. 4. Decision Tree Parameter Grid.

The best hyperparameters found for the DT were: 'criterion': 'entropy', 'max_depth': 10, 'max_features': 'log2', 'max_leaf_nodes': 10, 'min_samples_leaf': 1, 'min_samples_split': 10. The best model was retrained on the full training dataset and then evaluated on the testing set. The resulting DT is shown in Fig. 5.

SVM: This model was initialized using the Scikit-learn's SVC class with the default settings. Next, a grid search was performed over the values in the parameter grid shown in Fig. 6. The parameters tuned included: C (regularization parameter), coef0 (a term added to the polynomial kernel function that controls how much influence the higher degree terms have), degree (degree of the polynomial kernel function),

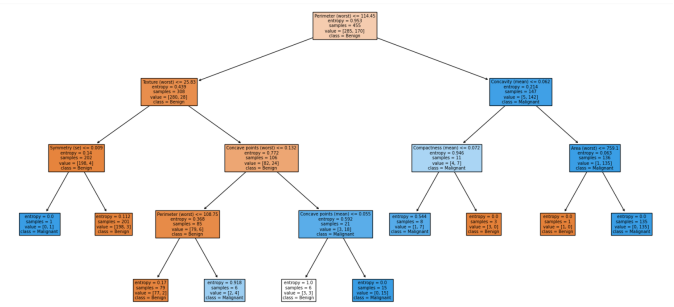

Fig. 5. Decision Tree

gamma (influence of a single training example), and kernel (type of kernel function used for transforming data into a higher-dimensional space).

```
'C': [0.01, 0.1, 0.2],
'gamma': [0.08, 0.09, 0.1, 0.11, 'scale'],
'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
'degree': [2, 3, 4],
'coef0': [1.6, 1.7, 1.8, 1.9]
```

Fig. 6. SVM Parameter Grid

The best hyperparameters found for the SVM model were: 'C': 0.01, 'coef0': 1.8, 'degree': 4, 'gamma': 0.09, 'kernel': 'poly'. The best model was retrained on the full training dataset and then evaluated on the testing set.

BLR: This model was initialized using Scikit-learn's BaggingClassifier class with the best Logistic Regression model found in the Logistic Regression grid search as the base estimator. Next, a grid search was performed over the values in the parameter grid shown in Fig. 7. The parameters tuned included: bootstrap (whether samples are drawn with replacement), max_features (maximum number of features used for training each base estimator), max_samples (fraction of samples used for training each base estimator), and n_estimators (number of base estimators in the ensemble).

```
'n_estimators': [10, 20, 30],
'max_samples': [0.5, 0.7, 1.0],
'max_features': [0.5, 0.7, 1.0],
'bootstrap': [True, False]
```

Fig. 7. Bagging Logistic Regression Parameter Grid

The best hyperparameters found for the Bagging Logistic Regression model were: 'bootstrap': True, 'max_features': 1.0, 'max_samples': 0.7, 'n_estimators': 20. The best model was retrained on the full training dataset and then evaluated on the testing set.

XGB: The XGB model was initialized using Scikit-learn's XGBClassifier class. Next, a grid search was performed over the values in the parameter grid shown in Fig. 8. The parameters tuned included: max_depth (maximum depth of a tree), learning_rate (amount by which the model's weights are adjusted during each step), n_estimators (number of boosting rounds), subsample (fraction of samples used per tree), colsample_bytree (fraction of features used per tree), and gamma

(minimum loss reduction required to make a further partition on a leaf node).

```
'n_estimators': [83, 84, 85],
'learning_rate': [0.08, 0.09, 0.1],
'max_depth': [4, 5, 6],
'subsample': [0.8, 0.9, 1.0],
'colsample_bytree': [0.2, 0.3, 0.4],
'gamma': [0.09, 0.1, 0.11]
```

Fig. 8. XGBoost Parameter Grid

The best hyperparameters found for the XGBoost model were: 'colsample_bytree': 0.2, 'gamma': 0.09, 'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 83, 'subsample': 1.0. The best model was retrained on the full training dataset and then evaluated on the testing set.

ASVM: This model was initialized using the Scikit-learn's AdaBoostClassifier class with the best Support Vector Machine (SVM) found in the SVM grid search as the base estimator. Next, a grid search was performed over the values in the parameter grid shown in Fig. 9. The parameters tuned included: n_estimators (the number of base estimators), learning_rate (controls the contribution of each base estimator), and algorithm ('SAMME' or 'SAMME.R').

```
'n_estimators': [50, 100, 200],
'learning_rate': [0.01, 0.1, 1.0],
'algorithm': ['SAMME', 'SAMME.R']
```

Fig. 9. AdaBoost Parameter Grid

The best hyperparameters found for the ASVM model were: 'algorithm': 'SAMME.R', 'learning_rate': 0.01, 'n_estimators': 100. The best model was retrained on the full training dataset and then evaluated on the testing set.

SE: This model was initialized using Scikit-learn's StackingClassifier class. The base models were all of the other models. The final classifier was a LR model. Next, a grid search was performed over the values in the parameter grid shown in Fig. 10. The parameters tuned included: C (regularization strength), penalty (L1 or L2 regularization), solver (liblinear, saga, sag, lbfgs), and max_iter (maximum number of iterations).

The best hyperparameters found for the Stacked Ensemble were: 'final_estimator__C': 1, 'final_estimator__max_iter': 100, 'final_estimator__penalty': 'l1', 'final_estimator__solver': 'liblinear'. The best model was retrained on the full training dataset and then evaluated on the testing set. The predictions from each model were used to train the final LR model, which made the final classification decision.

Overall, the techniques that were applied to improve the performance were: grid search with cross-validation, feature scaling with Scikit-Learn's StandardScaler, PCA (had no effect), ensemble methods, and L1 and L2 regularization. Grid search was used to find the best hyperparameters for each model. Feature scaling was used to balance the scale of all the features. This stops features with a larger scale from

```
    'final_estimator__C': [0.01, 0.1, 1, 10],
    'final_estimator__penalty': ['l1', 'l2'],
    'final_estimator__solver': ['liblinear'],
    'final_estimator__max_iter': [100]
},
{
    'final_estimator__C': [0.01, 0.1, 1, 10],
    'final_estimator__penalty': ['l1', 'l2'],
    'final_estimator__solver': ['saga'],
    'final_estimator__max_iter': [100000]
},
{
    'final_estimator__C': [0.01, 0.1, 1, 10],
    'final_estimator__penalty': ['l2'],
    'final_estimator__solver': ['lbfgs', 'sag'],
    'final_estimator__max_iter': [6000]
```

Fig. 10. Stacking Ensemble Parameter Grid

dominating the learning process. PCA was attempted to reduce the dimensionality of the data to avoid multicollinearity caused by high correlations between features. Ensemble methods were used in an attempt to increase the performance of individual models. L1 and L2 regularization were used with LR to penalize large coefficients so that those features do not dominate the learning process. Finally, stacking was used to combine all the models and use the unique strengths of each.

IV. Comparison

A comparison of the accuracy scores is shown in Fig. 11. The SE has the highest accuracy score, indicating it is the most effective model overall for this classification task. LR, BLR, and ASVM are fairly close to the highest scorer, indicating that these models are also highly effective. SVM and XGB showed slightly lower accuracies. The DT has the lowest score, and it is significantly lower than most of the models. It should be noted that applying Adaboosting to the SVM caused a significant increase in the accuracy score. However, applying bagging to LR made no difference to the accuracy score.

Ranking by Accuracy:

Model	Accuracy
Stacked Classifier	0.991
Logistic Regression	0.982
Bagging Logistic Regression	0.982
AdaBoosted SVM	0.982
XGBoost	0.956
SVM	0.956
Decision Tree	0.939

Fig. 11. Accuracy Score Comparison

A comparison of the mean cv scores is shown in Fig. 12. The BLR model has the highest Mean CV Score, indicating that this model will generalize well to unseen data, and it performs consistently well across the different folds of the data. However, the score for the SVM, SE, LR, and ASVM are very close to it. This shows that these models also perform reliably. The score for the XGB model is slightly lower. The DT had the lowest score, and it is significantly lower than the highest scoring model, indicating it is less reliable across different data folds. It should be noted that applying bagging to LR caused a small increase in the mean cv score. However, applying

Adaboosting to SVM actually caused a small decrease to the mean cv score.

Ranking by Mean CV Score:

Model	Mean CV Score
Bagging Logistic Regression	0.980
SVM	0.978
Stacked Classifier	0.978
Logistic Regression	0.976
AdaBoosted SVM	0.976
XGBoost	0.969
Decision Tree	0.943

Fig. 12. Mean CV Score Comparison

A comparison of the standard deviation of the cv scores is shown in Fig. 13. The BLR model had the lowest standard deviation of CV score, indicating it is the most consistent model in terms of performance across the different cross-validation folds. The ASVM, SE, SVM, and LR had slightly higher standard deviations, indicating a small increase in variability. The XGB and DT models have a much higher standard deviations, indicating that these models are more sensitive to variations in the data and they may not generalize as well as the other models to unseen data.

Ranking by Std CV Score:

Model	Std CV Score
Bagging Logistic Regression	0.013
AdaBoosted SVM	0.015
SVM	0.016
Stacked Classifier	0.016
Logistic Regression	0.019
XGBoost	0.025
Decision Tree	0.027

Fig. 13. CV Standard Deviation Comparison

A comparison of the recall for class 1 (malignant) is shown in Fig. 14. This score shows how well the models predict malignant cases. The SE performed the best. LR, BLR, and ASVM also performed well. XGB and SVM did not perform very well, and the DT performed the worst. Out of all of the metrics recall for class 1 is the most important for this classification problem, because the entire point of this problem is to diagnose breast cancer. Therefore, since the SE performs best here, it is the best model for this classification problem.

Ranking by Recall Class 1:

Model	Recall Class 1
Stacked Classifier	0.98
Logistic Regression	0.95
Bagging Logistic Regression	0.95
AdaBoosted SVM	0.95
XGBoost	0.88
SVM	0.88
Decision Tree	0.83

Fig. 14. Recall Comparison for Class 1

A comparison of the precision for class 0 (benign) is shown in Fig. 15. This score is the proportion of cases that were predicted as benign that were actually benign. High precision indicates that the model is not misclassifying malignant cases as benign.

A comparison of the f1 scores for class 0 (benign) is shown in Fig. 16. This metric represents the harmonic mean of precision and recall. It is a balanced measure of the model's accuracy in identifying benign cases.

Ranking by Precision Class 0:

Model	Precision Class 0
Stacked Classifier	0.99
Logistic Regression	0.97
Bagging Logistic Regression	0.97
AdaBoosted SVM	0.97
XGBoost	0.94
SVM	0.94
Decision Tree	0.91

Fig. 15. Precision Comparison for Class 0

Ranking by F1-Score Class 0:

Model	F1-Score Class 0
Logistic Regression	0.99
Bagging Logistic Regression	0.99
AdaBoosted SVM	0.99
Stacked Classifier	0.99
XGBoost	0.97
SVM	0.97
Decision Tree	0.95

Fig. 16. F1 Comparison for Class 0

A comparison of the f1 scores for class 1 (malignant) is shown in Fig. 17. This metric represents the harmonic mean of precision and recall. It is a balanced measure of the model's accuracy in identifying malignant cases.

Ranking by F1-Score Class 1:

Model	F1-Score Class 1
Stacked Classifier	0.99
Logistic Regression	0.98
Bagging Logistic Regression	0.98
AdaBoosted SVM	0.98
XGBoost	0.94
SVM	0.94
Decision Tree	0.91

Fig. 17. F1 Comparison for Class 1

Additionally, all of the models haves scores of 100% for recall class 0 (Benign) and precision class 1 (malignant). This means that for all the models every actual benign case was classified as benign, and every case that was predicted to be malignant was actually malignant.

To evaluate the effectiveness of our models, we compared our results with those presented in the study by Hridoy et al. (2024) [5]. Both their study and ours utilized the Breast Cancer Wisconsin (Diagnostic) dataset. In both studies, standard preprocessing steps like scaling (using StandardScaler) were applied. Our LR achieved an accuracy of 98.2%, and theirs achieved a higher accuracy of 99.42%. Our SVM achieved an accuracy of 95.6%, and theirs had an accuracy of 98.83%. Our XGB model had an accuracy of 95.6%, and theirs had an accuracy of 97.08%. Our Decision Tree had an accuracy of 93.9%, and theirs had an accuracy of 97.08%. These differences in accuracy can likely be attributed to differences between the ranges of values in the parameter grids used or differences in the preprocessing steps (the study does not go into explicit detail) They may have included in the data preprocessing steps outlier detection and removal and/or some sort of feature transfomation (as mentioned before these will be considered for future work).

During the review of these values, we noticed that some of them had changed when the code was migrated from Google Colab to Jupyter Notebook. For example, the accuracy score for XGB increased from 95.6% to 97.4%. We ran the exact

same code again in both environments, and confirmed they do produce different results. Addittionally, Jupyter Notebook will not output the information generated by the verbose parameter for GridSearchCV, but Google Colab will. This was our method of showing intermediary results in regards to hyperparameter tuning.

Their study used some models that we did not including KNN, RF, MLP, and NB. Our study also used some models that their study did not including: SE, ASVM, and BLR.

It is worth noting that our Stacked Classifier has nearly as high of an accuracy as the highest scoring model in Hridoy et al. (2024) [5]. Additionally, if the base models we used in our Stacked Classifier were as well tuned as the ones in the study, then the performance of the Stacked Classifier Would likely surpass the highest scoring model in their study. There is also the consideration of which models to use for the base models. A selection of a diverse group of the highest performing models from the table shown below would likely be a better predictor for this classification problem than the highest performing model (KNN) in their study. However, this would still include their KNN model as well as all of the other models shown in the table below.

The study by Hridoy et al. (2024) [5] also has a table showing the results of the highest scoring model from various recent studies. This is shown in 18. Given that this table is taken directly from this study, it should be noted that under the author column the row labeled "Our study" is not actually our results, instead they are the results of the study referenced. Again, our highest scoring model is the SE with an accuracy score of 99.1%.

Table 6 Comparison of the outcome with recent studies in the literature

Author	Model	Recall (%)	Accuracy (%)
Han and Yin [4]	ANN	98.41	98.74
Chen *et al.* [10]	XGB	100.0	97.40
Akkur *et al.* [11]	SVM	99.52	98.77
Naji *et al.* [12]	SVM	*NG*[1]	97.20
Gopal *et al.* [13]	MLP	97.00	98.00
Omondiagbe *et al.* [14]	SVM	98.41	98.82
Bensaoucha [15]	SVM, MLP	*NG*	96.52
Albadr *et al.* [16]	FLN	99.40	98.83
Ogundokun *et al.* [17]	MLP	97.80	97.20
Sukmandhani *et al.* [18]	H2O	89.62	93.14
Our study	k-NN	100.0	99.42

NG[1]: Not given

Fig. 18. Comparison of the performance of the best models from recent studies

The study by Kumar et al. [2] implemented a stacked ensemble that achieved an accuracy of 99.45%. Our Stacked Ensemble's score of 99.1% is not far behind this. Additionally, we could not find any studies that used ASVM or BLR on this dataset or similar datasets.

V. Future Directions

If we had more time there are many things we would add to this project to improve the performance. First, we would implement deep learning models, if we could get a larger amount of data. Then, we would include outlier detection and removal and also feature engineering in the preprocessing steps. Next, we would consider advanced feature selection techniques, such as Recursive Feature Elimination (RFE). Then, we would consider using Bayesian Optimization instead of grid search, because it is more efficient. This would allow us to search the hyperparameter space more thoroughly. Another option is to use different datasets and/or implement Synthetic Data Generation to artificially increase the size of the datasets. Finally, we would employ t-tests to statistically validate our evaluation of the models.

VI. Conclusion

This project focused on predicting breast cancer using the Breast Cancer Wisconsin (Diagnostic) dataset through the implementation of machine learning algorithms including: LR, DT, SVM, BLR, XGB, ASVM, SE. We then evaluated the effectiveness of the different models. The SE was the most effective model. This model achieved the highest accuracy score of 99.1%. It also achieved the highest recall for class 1, 98%.

The accuracy may seem great, and it is. However, this translates to essentially misdiagnosing a tumor in nearly 1 out of 100 cases (people). In this case, the stakes are very high. While our results are not far behind many other studies, the end goal should be to have a nearly perfect accuracy score. Still, this performance indicates that the SE approach, which combines the strengths of multiple models, is particularly well-suited for this problem.

As mentioned in the comparison section, the accuracy of the SE could be further increased by using the models that have shown the best results in recent studies as the base models. Additional efforts to make improvements to our solution could include: using Bayesian Optimization instead of grid search, implementing deep learning models such as Convolutional Neural Networks, performing additional data preprocessing steps, and obtaining a larger set of data or performing Synthetic Data Generation.

These enhancements might lead to even more accurate models, improving breast cancer diagnosis and potentially saving lives through earlier detection. Additional models that may be better suited to this problem include: Artificial Neural Networks, CNNs, MLPs, Feedforward Neural Networks, and KNNs.

References

[1] Wolberg,William, Mangasarian,Olvi, Street,Nick, and Street,W.. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. https://doi.org/10.24432/C5DW2B [Accessed: July 27, 2024]

[2] M. Kumar, S. Singhal, S. Shekhar, B. Sharma, and G. Srivastava, "Optimized stacking ensemble learning model for breast cancer detection and classification using machine learning," Sustainability, vol. 14, no. 21, p. 13998, 2022. Available: https://doi.org/10.3390/su142113998 [Accessed: July 27, 2024]

[3] American Cancer Society, "How common is breast cancer?," Available: https://www.cancer.org/cancer/types/breast-cancer/about/how-common-is-breast-cancer.html [Accessed: July 27, 2024]

[4] Y. Zhang, "Deep Learning in Wisconsin Breast Cancer Diagnosis," Towards Data Science, 2024. Available: https://towardsdatascience.com/deep-learning-in-winonsin-breast-cancer-diagnosis-6bab13838abd [Accessed: July 27, 2024]

[5] R. H. Hridoy, A. D. Arni, S. K. Ghosh, N. R. Chakraborty, and I. Mahmud, ”Performance enhancement of machine learning algorithm for breast cancer diagnosis using hyperparameter optimization,” ResearchGate, 2024. Available: https://www.researchgate.net/publication/377700718_Performance_enhancement_of_machine_learning_algorithm_for_breast_cancer_diagnosis_using_hyperparameter_optimization [Accessed: July 27, 2024]