Breast Cancer Classification Using Machine Learning

Hussain Sadiq Abuwala
School of Information Technology,
Monash University Malaysia,
hsabu1@student.monash.edu

***Abstract*—In medical diagnosis, the use of machine learning algorithms is growing gradually. This is primarily due to the fact that the efficacy of classification and recognition schemes has greatly increased to assist medical specialists diagnose illnesses. Such a disease is breast cancer, a sort of cancer that is very prevalent among women. In this paper, breast cancer diagnosis was conducted using 4 different machine learning algorithms. The evaluation methods used were accuracy, precision, recall and others. Among the 4 classification algorithms, random forest performed the best with an accuracy of 99.3%. Hence, the obtained results show that random forest can be used as an effective tool for diagnosing breast cancer.**

***Index Terms*—classification, machine learning, breast cancer, support vector machines, random forests, k-nearest neighbor, decision trees.**

I. Introduction

CANCER is the uncontrolled growth of abnormal cells in the body. It is usually named after the portion of the body in which it arose. Hence, breast cancer relates to the uneven development of cells in the breast tissue. A cluster of rapidly dividing cells can create an additional tissue lump or mass. These masses are referred to as tumors. Tumors can be cancerous (malignant) or cancer-free (benign). Malignant tumors enter healthy body tissues and kill them. The term, breast cancer, refers to a malignant tumor that has developed from cells in the breast. In females between the ages of 40 and 55, breast cancer is the major cause of death. The amount of instances of breast cancer has increased considerably in latest years. Breast cancer is revealed to be the second of the most identified cancers in [1]. Breast cancer is also reported to have been the most common cancer in the globe by 2002.

With the growth of more efficient diagnostic methods and changes in therapy methodologies, the results of breast cancer have enhanced over the past century. Early identification and precise diagnosis of this disease are the main variables in this trend. For females who have not metastasized breast cancer, the long-term survival rates has risen, with most females surviving many years after diagnosis and treatment [2]. In medical diagnosis, the use of classifier technologies is growing gradually. There is no question that the most important factors in diagnosis are the assessment of patient information and expert choices. But expert systems and various machine learning classification methods can also greatly assist professionals.

II. Literature Review

Medical images of breast biopsies contain a great deal of complex data and interpreting them can be very subjective. The line between a person having cancer and not having cancer is very thin and it can get very challenging to identify these thin margins. Sometimes, doctors do not even agree with their previous diagnosis when they are shown the same case a year later [3]. Machine Learning could provide more accurate classification consistently because by drawing from a large data set, the system can recognize patterns in the samples that are associated with cancer but are difficult for humans to see. Also automatic classification can assist pathologists by providing second opinions and reducing their workload. Automatic classification can also help save time for the doctors. In the medical field, doctors assign a text to each image and save in their database for future reference. Using automatic classification, the system will automatically assign text to new images during classification.

III. Methodology

All the steps taken to classify breast cancer are summarized in figure 1 which can be found on the next page. First subsection includes information regarding the dataset used for classification. Next two subsections talks about the data pre-processing steps and the training phase.

A. Dataset

The dataset used in this research is accessible to the public and was developed by Dr. William H. Wolberg, a physician at the University of Wisconsin Hospital in Madison, Wisconsin, USA. The dataset can be downloaded from the Kaggle website [4]. Dr. Wolberg used fluid samples from individuals with strong breast masses and an easy-to-use graphical computer program called Xcyt to produce the dataset. Xcyt is capable of analyzing cytological characteristics based on digital scans. It uses a curve-fitting algorithm to calculate ten features from each of the sample cells. After that it calculates the mean value, extreme value, and standard error of each feature for the image, returning a real-valued vector of 30. The ten features calculated are:

1) radius (mean of distances from center to points on the perimeter)
2) texture (standard deviation of gray-scale values)

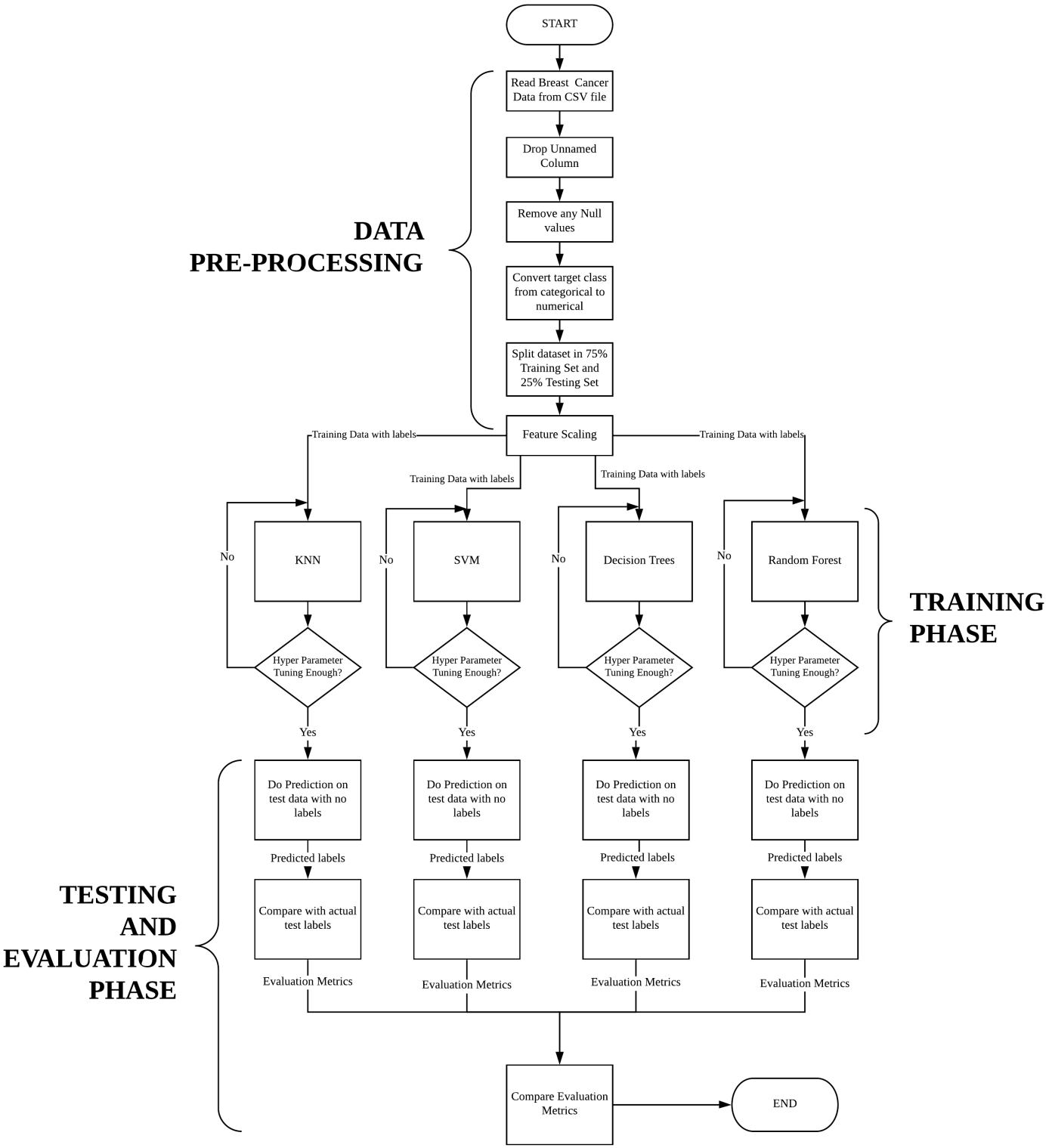

Fig. 1: Flowchart showing the method used for classification

3) perimeter
4) area
5) smoothness (local variation in radius lengths)
6) compactness
7) concavity (severity of concave portions of the contour)
8) concave points (number of concave portions of the contour)
9) symmetry
10) fractal dimension ("coastline approximation" — 1)

The dataset also contains an id column which basically represents individual patients. There is a diagnosis column which represents if the patient has breast cancer or not. Values of "M" representing malignant and "B" representing benign are used in the diagnosis column. The class distribution is 357 benign patients and 212 malignant patients. In summary, the dataset contains 32 columns and 569 rows where out of the 32 columns, there are 30 feature columns, 1 patient id column and lastly the diagnosis column which is basically the target class used for classification.

B. Data Preprocessing

The dataset is first read from the csv file as a data frame using the panda's library. An unnamed column is automatically added when reading the file, so it is removed using the "drop" command. Then the dataset is checked for any missing or null values. There were no missing values in this dataset. The target column in this dataset is categorical having the values "M" and "B". This represents a tumor being either malignant (cancerous tumor) or benign (non-cancerous tumor). As the inputs for machine learning algorithms are expected to be numerical, the categorical values for the diagnosis column are assigned the values 1 and 0 where 1 represents "M" and 0 represents "B" using the "LabelEncoder" library of scikit learn. To split the dataset for training and testing, scikit learn's "train_test_split" function is used. The dataset was split into 75% for training and 25% for testing.

Scaling is done next to bring the data in the feature columns of both the training and test sets onto a common scale. After inspecting the data, it is seen that the values in feature columns vary a lot. For example the mean of the column named "area_mean" is 654 and the mean of the column named "compactness_mean" is 0.1. But since, most of the machine learning algorithms use Euclidean distance between two data points in their computations, we need to bring all features to the same level of magnitude. To do this, scikit learn's "StandardScaler" library is used [5]. The math equations that this library uses are shown below:

$$\text{Standardization: } z = \frac{x - \mu}{\sigma} \tag{1}$$

$$\text{with Mean: } \mu = \frac{1}{N}\sum_{i=1}^{N}(x_i) \tag{2}$$

$$\text{and Standard Deviation: } \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{3}$$

The idea behind "StandardScaler" is that it will transform the data such that its distribution will have a mean value of zero and standard deviation of one using equations 2 and 3 . Given the distribution of the data, each value in the dataset will have the sample mean value subtracted, and then divided by the standard deviation of the whole dataset using the equation 1.

C. Training Phase

After the data-preprocessing step is finished, the training set is used to train different machine learning algorithms. The following subsections show how different machine learning algorithm works and how the hyper parameters of each algorithm were tuned to achieve better results.

1) K-Nearest Neighbours (KNN): KNN is a machine learning algorithm that can be used for both classification and regression. For the purpose of this research, KNN is used as a classification technique. The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these [6]. The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning). The distance can, in general, be any metric measure: standard Euclidean distance and Minkowski distance are the most common choices. Despite its simplicity, nearest neighbors has been successful in a large number of classification and regression problems, including handwritten digits and satellite image scenes. Scikit learn has a library function called "KNeighborsClassifier" for using this algorithm [7]. The hyper parameters that are avaialble to use for this function are:

- **n_neighbors:** number of neighbours to be taken into consideration for the voting process.
- **p:** power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan distance (l1). For p = 2, it is equivalent to using euclidean distance (l2). For arbitrary p, minkowski distance (l_p) is used. Equation 4 and 5 below shows how Minkowski distance is calculated:

The Minkowski distance of order p between two points

$$X = (x_1, x_2, ..., x_n) \text{ and } Y = (y_1, y_2, ..., y_n) \tag{4}$$

is defined as

$$D(X, Y) = (\sum_{i=1}^{N} |x_i - y_i|^p)^{\frac{1}{p}} \tag{5}$$

Firstly, "n_neighbors" parameter tuning was performed. AUC curve was plotted for different values of "n_neighbors" for both the training and testing set. As shown in figure 2,

AUC score for the testing data (red line) is highest when the neighbour value is 15. So the optimal neighbour value is 15. Keeping "n_neighbors" at a constant value of 15, "p" parameter tuning was performed. An AUC curve was plotted for different values of "p" for both the training and testing set. As shown in figure 3, AUC score for the testing data (red line) is highest when the p value is 2. Final parameter values for "n_neighbors" and "p" are 15 and 2 respectively. For "n_neighbors" tuning, integer values between 1 and 30 were tested. For "p" tuning, integer values between 1 and 5 were tested.

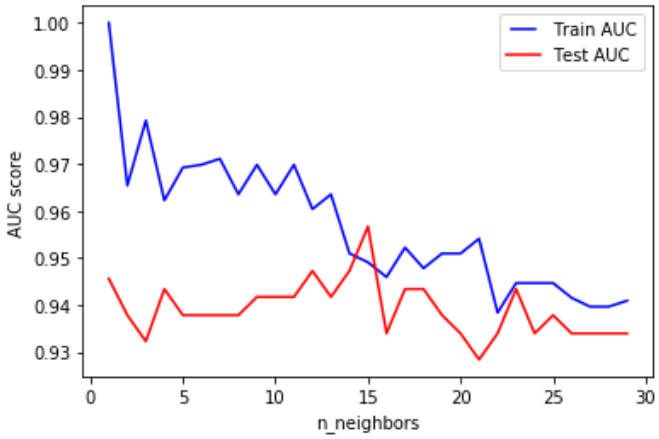

Fig. 2: n_neighbour parameter tuning (KNN)

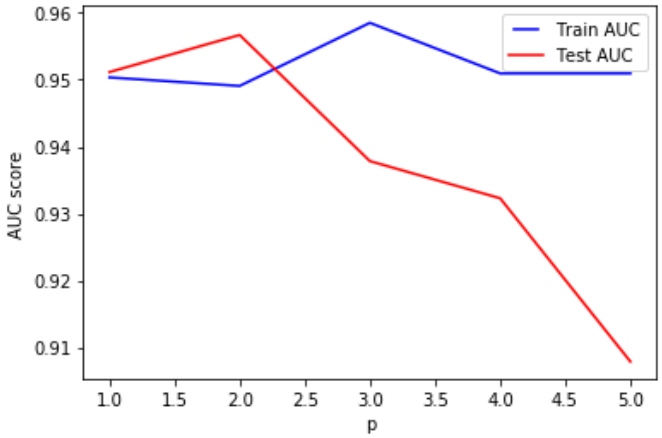

Fig. 3: p parameter tuning (KNN)

2) Decision Trees (DT): Decision tree learning is the construction of a decision tree from class-labeled training tuples. A decision tree is a flow-chart-like structure, where each internal (non-leaf) node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf (or terminal) node holds a class label. The topmost node in a tree is the root node. Decision tree generation consists of two phases: 1) Tree construction and 2) Tree Pruning. In the first phase, all the training examples are at the root. Examples are partitioned recursively based on selected attributes. This partition is done by choosing a variable at each step that best splits the set of examples. Different algorithms use different metrics for measuring "best". These algorithms generally measure the homogeneity of the target variable within the subsets. In the second phase, tree branches that reflect noise or outliers are identified and removed. Scikit learn has a library function called "DecisionTreeClassifier" for using this algorithm [8]. The hyper parameters that are avaialble to use for this function are:

- **max_depth:** indicates how deep the tree can be.
- **min_samples_split:** represents the minimum number of samples required to split an internal node.
- **max_features:** represents the number of features to consider when looking for the best split.

Firstly, "max_depth" parameter tuning was performed. An AUC curve was plotted for different values of "max_depth" for both the training and testing set. As shown in figure 4, AUC score for the testing data is highest when "max_depth" is equal to 4. So "max_depth" value of 4 is selected as optimal. Keeping "max_depth" at a constant value of 4, "min_samples_split" parameter tuning was performed. An AUC curve was plotted for different values of "min_samples_split" for both the training and testing set. As shown in figure 5, AUC score for the testing data is highest when "min_samples_split" is equal to 0.05. So "min_samples_split" value of 0.05 is selected as optimal.

Keeping "max_depth" and "min_samples_split" at a constant value of 4 and 0.05, "max_features" parameter tuning was performed. An AUC curve was plotted for different values of "max_features" for both the training and testing set. As shown in figure 6, AUC score for the testing data is highest when "max_features" is equal to 15. So "max_features" value of 15 is selected as optimal. Final parameter values for "max_depth", "min_samples_split" and "max_features" are 4, 0.05 and 15 respectively. For "max_depth" tuning, integer values between 1 and 32 were tested. For "min_samples_split" tuning, decimal values between 0.1 and 1.0 were tested to represent 10% to 100% of the samples. For "max_features" tuning, integer values between 1 and 30 were tested.

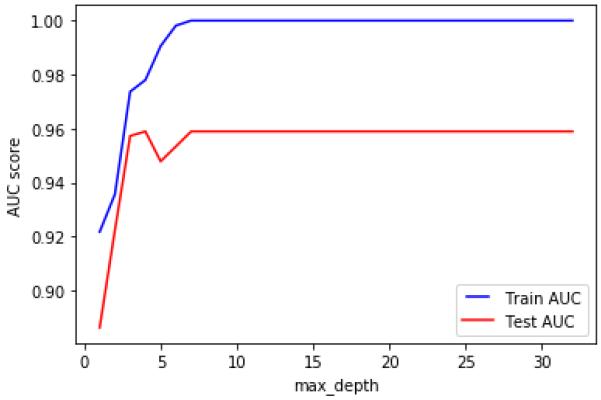

Fig. 4: max_depth parameter tuning (DT)

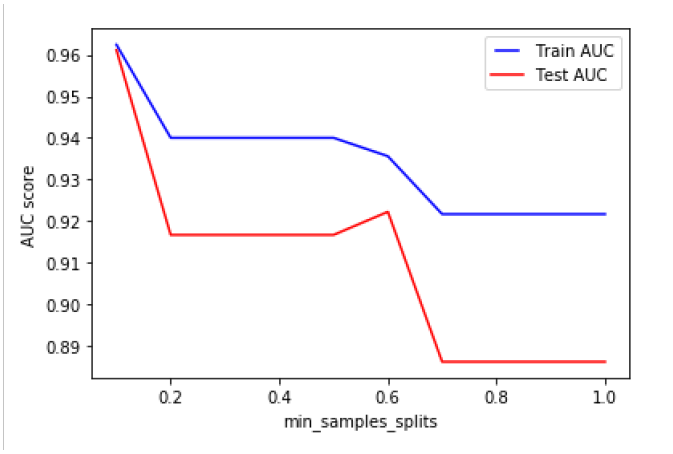

Fig. 5: min_samples_split parameter tuning (DT)

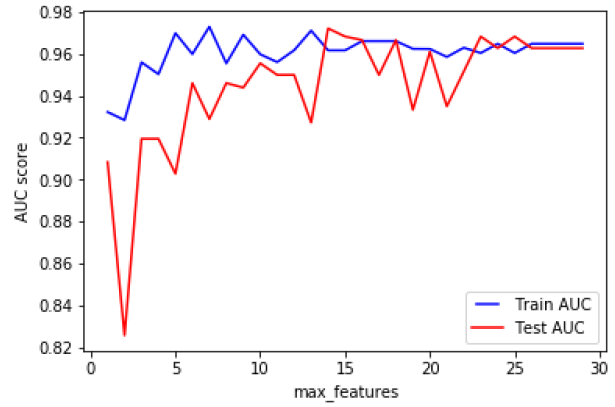

Fig. 6: max_features parameter tuning (DT)

3) Random Forests (RF): Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. In this research, random forests is used for classification. Scikit learn has a library function called "RandomForestClassifier" for using this algorithm [9]. The hyper parameters that are avaialble to use for this function are:

- **N_estimators:** represents the number of trees in the forest
- **max_depth:** represents the depth of each tree in the forest.
- **max_features:** represents the number of features to consider when looking for the best split.

Firstly, "N_estimators" parameter tuning was performed. An AUC curve was plotted for different values of "N_estimators" for both the training and testing set. As shown in figure 7, it can be seen that the test accuracy is highest when "N_estimators" is equal to 10. So the "N_estimators" value of 10 is selected as optimal. Keeping "N_estimators" at a constant value of 10, "max_depth" parameter tuning was performed. An AUC curve was plotted for different values of "max_depth" for both the training and testing set. As shown in figure 8, it can be seen that the test accuracy is highest when "max_depth" is equal to 5. So the "max_depth" value of 5 is selected as optimal. Keeping "N_estimators" and "max_depth" at a constant value of 10 and 5 respectively, "max_features" parameter tuning was performed. An AUC curve was plotted for different values of "max_features" for both the training and testing set. As shown in figure 9, it can be seen that the test accuracy is highest when "max_features" is equal to 27. So the ""max_features" value of 27 is selected as optimal. Final parameter values for "N_estimators", "max_depth" and "max_features" are 10, 5 and 27 respectively. For "N_estimators" tuning, integer values tested are

$$1, 2, 4, 8, 16, 32, 64, 100, 200$$

For "max_depth" tuning, integer values between 1 and 32 were tested. For "max_features" tuning, integer values between 1 and 30 were tested.

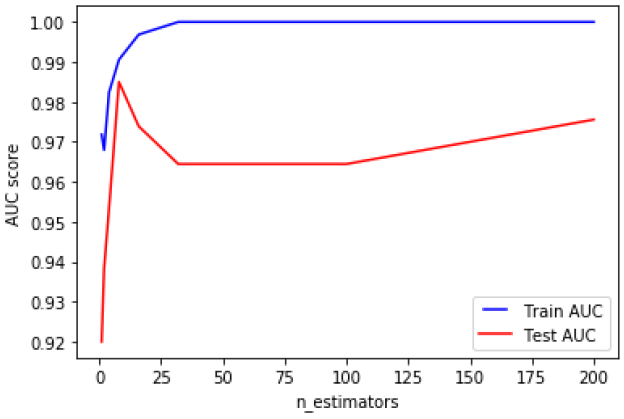

Fig. 7: n_estimators parameter tuning (RF)

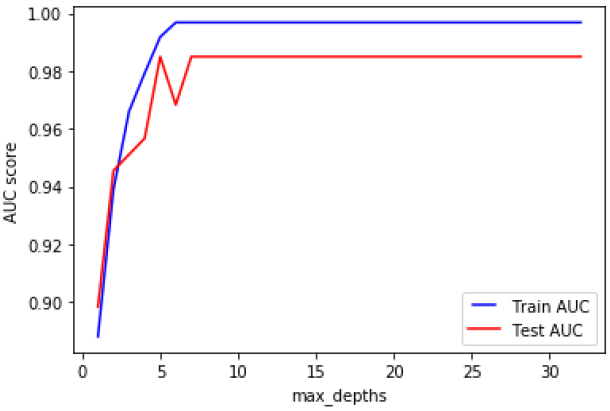

Fig. 8: max_depth parameter tuning (RF)

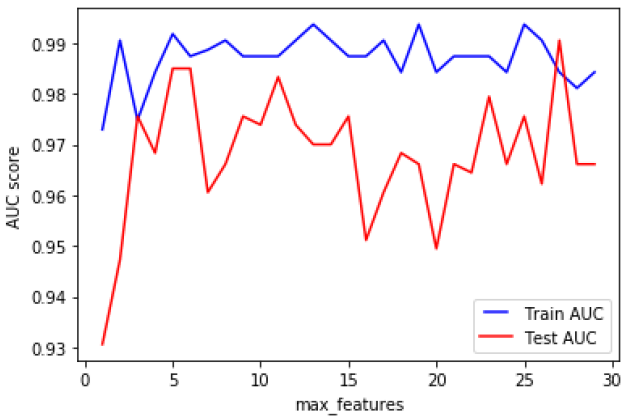

Fig. 9: max_features parameter tuning (RF)

4) Support Vector Machines (SVM): A support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Given a set of training examples, each marked as belonging to one or the other of two categories, a SVM training algorithm builds a model that assigns new examples to one category or the other. A SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall. Scikit learn has a library function called "SVC" for using this algorithm [10]. The hyper parameters that are avaialble to use for this function are:

- **C:** is the penalty parameter of the error term. It controls the trade off between smooth decision boundary and classifying the training points correctly.
- **gamma:** is a parameter for non linear hyperplanes. The higher the gamma value, the more it tries to exactly fit the training data set.
- **kernel:** selects the type of hyperplane used to separate the data. Using 'linear' will use a linear hyperplane (a line in the case of 2D data). 'rbf' and 'poly' uses a non linear hyper-plane

As there are many possible combinations of parameters for SVM, writing code manually would be a time consuming process. Using the scikit learns "GridSearchCV", it is possible to do all combinations of parameter values provided and the best parameter combination can easily be found. The "GridSearchCV" function needs input of values of the parameters that it needs to test. Firstly, "C" parameter values that are given to the "GridSearchCV" function are 0.001, 0.01, 0.1, 1, and 10. Secondly, "gamma" parameter values that are given to the "GridSearchCV" function are 0.001, 0.01, 0.1, and 1. Finally, "kernel" parameter values that are given to the "GridSearchCV" function are "linear", "rbf" and "poly". The best combination of parameter values given by the GridSearchCV were 0.1, 0.001 and linear for "C", "gamma" and "kernel" respectively.

IV. Results and Discussion

Each of the trained models were tested with the same test data. Different evaluation metrics were extracted from each of the trained models. Evaluation metrics that were used were:

- **True Positive (TP):** number of times both the actual and the predicted class values are "malignant".
- **False Positive (FP):** number of times the actual class is "malignant" and the predicted class is "benign".
- **True Negative: (TN)** number of times both the actual and the predicted class values are "benign".
- **False Negative (FN):** number of times the actual class is "benign" and the predicted class is "malignant".
- **Precision:** when the model predicts the class value to be "malignant", how many times is it correct?. Calculated using the formula $TP/(TP + FP)$
- **Recall:** when the actual class value is "malignant", how many times does the model predict the class to also be "malignant"? Calculated using the formula $TP/(TP + FN)$
- **Accuracy:** number of correct predictions by the model from the total number of test data. Calculated using the formula $(TP + TN)/(TP + FP + FN + TN)$

	KNN	DT	SVM	RF
Accuracy before parameter tuning	0.950	0.880	0.90	0.972
Accuracy after parameter tuning	0.965	0.965	0.965	0.993

TABLE I: Accuracy of classifiers before and after tuning

	KNN	DT	SVM	RF
TP	49	52	50	52
FP	1	4	2	0
TN	9	86	88	90
FN	4	1	3	1
Precision	0.98	0.93	0.96	1
Recall	0.92	0.98	0.94	0.98

TABLE II: Comparison of Fine-Tuned Classifiers using Evaluation metrics

All the evaluation metrics were recorded in both TABLE I and TABLE II. In terms of improvement of accuracy before and after parameter tuning, from TABLE I it can be seen that decision trees have the highest accuracy improvement of 8.5%. In terms of precision, from TABLE II it can be seen that all of the models were able to give very good results($>= 0.93$). Among them random forest was able to

perform best with a perfect precision score of 1. KNN had the second best precision score with a value of 0.98. In terms of recall, from TABLE II it can be seen that all of the models were again able to give very good results($>= 0.92$). Among them random forest and decision trees were tied with a near perfect recall score of 0.98. In terms of accuracy of models after fine tuning, from TABLE I it can be seen that random forest was able to perform best with a near perfect accuracy score of 0.993. KNN, Decision trees and SVM were second best with an accuracy score of 0.965. In terms of time taken for parameter tuning, KNN took the least time. As there is only one parameter that had to be tuned for KNN (neighbors parameter) compared to 3 parameters for the other three, it took lesser time to find the optimal parameter for KNN.

It is easier to analyze why random forests provides better results compared to decision trees [11]. There are two reasons mainly. Firstly, there is a reduction in overfitting by averaging several trees. Secondly, there is less variance using multiple trees. This means the chance of stumbling across a classifier that doesn't perform well because of the relationship between the train and test data is less when using multiple trees. Performance of KNN and SVM depend heavily on the dataset itself. To further improve the accuracy of both KNN and SVM, the dataset needs to be studied further to see how separable the data points are in different dimensional space.

V. Conclusion

With the improvements in machine learning tools, the effects of these innovations are entering to more application domains day-by-day and medical field is one of them. Decision making in medical field can be a trouble sometimes. Classification systems that are used in medical decision making provide medical data to be examined in shorter time and more detailed. According to the statistical data for breast cancer in the world, this disease is among the most prevalent cancer types. In the same time, this cancer type is also among the most curable ones if it can be diagnosed early. In this study, for the diagnosis of breast cancer, different machine learning classification algorithms were compared on how well they could classify unseen breast cancer data. Among the classification algorithms, random forest was able to provide the best results with an accuracy of 99.3%. The results strongly suggest that random forest can aid in the diagnosis of breast cancer. This research also has some limitations. Firstly, the splitting statergy used for getting the training and test data could be bias. It could be possible that a classifier that performs well on some specific test set may not perform that well on other test sets. So it would be better to remove this bias by using K-fold cross validation technique. Another limitation is that the hyper-parameter that were selected as optimal may only be a local optimal compared to being a global optimal because of the fact that the ranges that were used to test each hyper-parameter were defined manually. For future research, I would like to focus on using K-fold cross validation technique for testing my classification models and also use more advanced hyper-parameter tuning strategies. Furthermore, the use of other classification algorithms like neural networks can also be explored in classifying breast cancer.

References

[1] (2019, July). [Online]. Available: https://www.wcrf.org/dietandcancer/cancer-trends/worldwide-cancer-data

[2] D. West, P. Mangiameli, R. Rampal, and V. West, "Ensemble strategies for a medical diagnostic decision support system: A breast cancer diagnosis application," *European Journal of Operational Research*, vol. 162, no. 2, pp. 532–552, 2005.

[3] E. Mercan, S. Mehta, J. Bartlett, L. G. Shapiro, D. L. Weaver, and J. G. Elmore, "Assessment of machine learning of breast pathology structures for automated differentiation of breast cancer and high-risk proliferative lesions," *JAMA network open*, vol. 2, no. 8, pp. e198 777–e198 777, 2019.

[4] (2016, September). [Online]. Available: https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

[5] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[6] [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html

[7] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[8] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[9] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[10] [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[11] N. Horning, "Introduction to decision trees and random forests," *Am. Mus. Nat. Hist*, vol. 2, pp. 1–27, 2013.