Using Artificial Intelligence Through a Machine Learning Algorithm (CART) to Predict the Malignancy of a Breast Tumor

Aniruddha Murali

Abstract— Breast cancer is responsible for causing the greatest number of cancer-related deaths among women, impacting 1.5 million women every year (WHO). One way to reduce the number of deaths caused by breast cancer is to perform early diagnosis to detect the presence of a malignant tumor before the tumor gets too harmful. While there are several methods of diagnosing and testing a tumor, they all have their own sets of problems: they are time-consuming, expensive, and limited in their ability to diagnose a variety of tumors. In this study, a machine learning algorithm was developed to predict if someone has breast cancer. The diagnostic model uses ten different parameters received from a mammography, a screening, or an ultrasound test of the breast tumor to output its prediction. The model returns whether or not the tumor is malignant or benign and the accuracy of the prediction. Prediction accuracy was between 90 and 93%. In addition to machine learning, the Point-Biserial Correlation Coefficient was used in conjunction with a t-test to examine the strength of association between each of the ten features and malignancy. An app was that utilized the machine learning algorithm was developed to provide breast cancer specialists with a user interface for the diagnosis model. By using machine learning to assess a breast tumor, there will be a rapid, non-invasive, and inexpensive way to detect breast cancer. This algorithm can help thousands of women get early treatment and provide a way to address any type of binary classification problem.

Index Terms— breast cancer, binary classification, decision tree, machine learning, prediction, malignancy, Point-Biserial Correlation

1 Introduction

Breast cancer has been a prominent issue among women; it has been very difficult to determine whether a breast tumor is malignant or not without being invasive and to address the cancer before it becomes too problematic. Cancers can result from alterations in genes encoding cellular signaling molecules, especially protein kinases (My Cancer Genome). Types of gene alterations that can result in cancers include: single nucleotide variants (point mutations), small duplications of consecutive nucleotides, insertions or deletions involving one or a few nucleotides, changes in exon or gene copy numbers, and structural variants in genetic material including translocations and inversions (My Cancer Genome).

Current methods of tumor testing can identify mutations in tumor DNA. However, these methods generally come with a set of drawbacks. Almost all current methods of tumor testing can only detect a specific mutation; other mutations that may be present in tumor DNA cannot be detected (My Cancer Genome). Such methods are limited in the types or number of mutations that can be detected in tumor DNA. They can also be labor intensive and/or expensive, often involving the use of highly sophisticated technology (My Cancer Genome). Completion of diagnosis can take anywhere from 2-3 days to several weeks, and such tests also have a possibility of false negatives and false positives (My Cancer Genome).

• *Aniruddha Murali is actively working on scientific research in the field of artificial intelligence as a junior at Staples High School, Westport, CT, USA, PH-12034518797. E-mail: aniruddha.murali@gmail.com*

Early diagnosis of breast tumors can help doctors to provide a mostly accurate assessment of a breast tumor to their patients. The most common method of diagnosing breast tumors is mammography, which is an x-ray imaging method used to examine a breast for early detection of breast cancer (NIBIB). A radiologist examines a mammogram to identify any potential abnormalities in the breast. Mammography has been shown to reduce breast cancer mortality by about 20% in high-resource settings (WHO). However, a mammogram is examined only by a radiologist. If it is not clear that the tumor is malignant or benign, there is a chance that the radiologist could give an inaccurate result. In one study, 100 mammograms were submitted to nine radiologists. The diagnosis or suspicion of cancer varied from 10-55% (Devitt). The denser the breast, the more difficult it is to produce an image, so it will be harder to diagnose (NIBIB). Mammograms also use x-rays, which means that patients are exposed to some radiation.

The concept of machine learning has expanded in use into multiple fields, and it can be used to provide an early and precise diagnosis of a breast tumor. In this project, a program was developed to be able to use de-identified data that has already been verified by the University of Wisconsin to create an algorithm that outputs the malignancy of the tumor and the accuracy of the prediction. This program creates a decision tree to predict if a tumor is malignant or benign. There are multiple advantages of using a decision tree algorithm. Decision trees are simple and easy to interpret. They require very little data preparation; unlike other machine learning algorithms, decision tree do not require normalization of data, creation of dummy variables, or removal of blank values. The cost of a using a decision tree is logarithmic in the number of data points used to train the tree (Scikit Learn). This means that the

more data points used, the more accurate the prediction is. Decision trees can handle both numerical and categorical variables. Decision trees also use a white-box model. Black-box algorithms utilize unknown mechanisms behind the scenes, whereas white-box algorithms are completely transparent. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic (Scikit Learn). Decision trees are among the most accurate machine learning algorithms. The advantages of decision trees make them an appropriate algorithm to use for predicting the malignancy of breast tumors. Data collected from imaging of breast tumors can be used in a machine learning algorithm that utilizes decision trees to provide a prediction of the malignancy of a breast tumor.

2 METHODS

This project will use the Python programming language to develop a decision tree model that will predict whether or not someone who has had an ultrasound, mammogram, or MRI breast exam, has breast cancer. The model will utilize data from the University of Wisconsin. Each data point from the data set incorporates ten types of real-valued features that were computed for each breast mass: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension. The mean values of these features were computed for each tumor by the, resulting in 10 types of attributes for each data point. The data will be inputted into the program, which will then analyze the data and use the Classification and Regression Tree (CART) algorithm to determine if a tumor is malignant or benign. The objective is that a user should be able to input the ten features mentioned and get a result. For someone to be able to use the program to determine whether the tumor is malignant or benign, the patient should get a mammogram screening or ultrasound test to identify any abnormalities in the breasts and to measure the values of the ten types of features. The model could then use those ten types of attributes as an input and produce results that will identify the cells as malignant or benign and provide an accuracy of that prediction. A statistical analysis utilizing Point-Biserial Correlation Coefficients and t-tests could show which parameters actually have an association with malignancy of a breast tumor and how strong that association is (Fig. 4).

2.1 The Classification and Regression Tree (CART) Algorithm

CART constructs a binary decision tree in which one parameter is checked at each node of the tree. CART is used to perform a prediction of the classification of an object based on the object's properties. Decision trees split the nodes of all available variables based on a target variable that will help determine the best split from that node. The nodes split until they reach a terminal node. A row of data can be inputted into this decision tree, and when it reaches a terminal node, a final prediction is returned and printed on the screen. The prediction returned from a terminal node in the CART algorithm is a categorical variable. In this decision tree model, the categories for the class values are 0 and 1, where 0 is a prediction for a benign tumor and 1 is a prediction for a malignant tumor.

2.2 Gini Index

Gini index is the cost function used to evaluate splits in the dataset. A split in the dataset involves one input attribute (one explanatory variable) and one value (response variable) for that attribute. It can be used to divide training patterns into two groups of rows, where each group represents a list of values of one attribute. The Gini score shows how good a split is in one attribute by how mixed the class values are in the split of that attribute. A Gini score of 0 indicates a perfect split.

The first step to calculating the Gini index is to calculate the proportion of class values in each group:

$$proportion = \frac{frequency\ of\ class\ value}{number\ of\ rows} \quad (1)$$

Since there are only two possible class values, the sum of the proportion of group values that belong to class 0 and the proportion of group values that belong to class 1 is equal to 1. So, 1 minus the sum of all the squared proportions will yield the Gini index. Gini index is calculated for n values in the group:

$$gini\_index = 1 - \sum_{i=1}^{n} proportion_i^2 \quad (2)$$

This Gini index value for each group must then be weighted by the size of the group. If a group has a larger size than other groups, it will have more influence on the overall Gini index. The calculation is done as follows:

$$gini\_index = (1 - \sum_{i=1}^{n} proportion_i^2) \times \frac{group\ size}{total\ number\ of\ sampes} \quad (3)$$

Finally, the Gini scores from each of k groups are added to give a final Gini index for a potential split point:

$$gini\_index = \sum_{j=1}^{k} ((1 - \sum_{i=1}^{n} proportion_i^2) \times \frac{group\ size}{total\ number\ of\ sampes}) \quad (4)$$

This Gini index value can be used to evaluate a split. The closer it is to 0, the stronger the split is (0 is a perfect split). The *calculateGiniIndex()* (Fig. 7) function can calculate the Gini index given the groups and class values of those groups. This function is used to evaluate multiple different splits for a node in the decision tree; whichever group has the Gini index value closest to 0 will be set as a split point (node) in the decision tree.

2.3 Node Split

In order to form each node on the decision tree, all possible splits of the data for that decision node must be evaluated to see which split would be best for that particular node. This

uses the *calculateGiniIndex()* function to find the strength of a split based on the split point. The program must check all possible splits to determine which split yields the lowest Gini index. The *splitGroup()* function takes in three parameters: index of the attribute for the split, value of that attribute, and the dataset. It returns two arrays that are split from the dataset called "left" and "right." "left" consists of all rows of data in which the attribute value in that row was less than the parameter value while "right" consists of all rows of data in which the attribute value in that row was greater than or equal to the parameter value. The *getSplitBestNode()* (Fig. 7) function takes the dataset as a parameter and calculates the Gini index of all possible splits, which are found using the *splitGroup()* function. It returns the split which has the Gini index value closest to 0 in the form of a dictionary, which contains the index, value, and group separation of that split.

2.4 Terminal Nodes

When building a decision tree, terminal nodes must be created to form a final prediction. Terminal nodes must be put in place to prevent overfitting. If a decision tree is to large and complex, it will fit the original dataset very well, but will falter once additional data is given to it. To deal with overfitting, conditions must be set to limit the size and complexity of the decision tree. This model has two conditions in which it labels a node in the decision tree as a terminal node. The first condition is that the tree has a maximum depth; if the tree reaches a certain depth, the node at the maximum depth will become a terminal node. The value of the maximum depth is stored in the max_depth global variable. The second condition is that each node has a minimum number of training patterns it is responsible for checking; if a node has lower than the minimum number of training patterns, that node will become a terminal node. The value of the minimum number of training patterns is stored in the min_size global variable. After checking these two conditions, the model must create a terminal node and make a prediction. This is done in the *toTerminalNode()* function, which checks a group of rows and returns the most common class value. This class value is the prediction; if the value is 0, the model predicts that the tumor is benign, and if the value is 1, the model predicts that the tumor is malignant.

2.5 Bulding the Decision Tree

The first step to creating a decision tree is to create the root node and determine its split. The *getSplitBestNode()* function is first performed on the entire dataset to create the root node and to find the split which yields the lowest Gini index value. From there, recursion is used to create the decision nodes. The recursive process is done in the *splitNode()* (Fig. 7) function. The *splitNode()* function makes splits for a decision node or calls the *toTerminalNode()* function to create a terminal node. If the tree has reached its maximum depth or the node does not have at least the minimum number of required training patterns to check, the *splitNode()* function calls on *toTerminalNode()* to make a terminal node. Otherwise, *splitNode()* calls the *getSplitBestNode()* function to find the best split for that node, and then the *splitNode()* function is called recursively with the new left and right decision nodes created. This tree serves as the model for which predictions are made.

2.6 Prediction

The final step in classification is to predict if the tumor is malignant or benign. Given a row of data as an input, the *predict()* function uses the *splitNode()* function to navigate through the decision tree. Just like the *splitNode()* function, *predict()* uses recursion by calling itself with the new decision node. If the node has a dictionary, there is still another level of the decision tree, so *predict()* continues the recursion process. Otherwise, *predict()* can tell that it has reached a terminal node in the tree and will return the prediction.

2.7 Prediction Accuracy

Accuracy was calculated to determine the reliability of the CART algorithm used for the decision tree model. To calculate accuracy, the *calculateAccuracy()* function checks the prediction for each row of testing data with the actual result from that row. *calculateAccuracy()* returns the number of correct predictions divided by the total number of predictions:

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions} \quad (5)$$

Cross validation was used to give a better measurement of accuracy. The data was split into separate subsets. The number of subsets was stored in the n_data_sets global variable. The *testPredictiveModel()* (Fig. 7) function loops through a list of these subsets. In every iteration, one subset was used as the test sample, while the remaining subsets were used as training samples for the decision tree model. This ensures that variations within the whole dataset can be accounted for when calculating the accuracy of the model. Even though we know the results of the test samples, those results are not used in the model, so there is no bias introduced. The accuracies from each iteration were stored in the scores array. *testPredictiveModel()* returns the mean accuracy of all the subsets.

2.8 Statistical Analysis Using the Point-Biserial Correlation Coefficient and a t-test

Statistical analysis was used to investigate which of the ten cellular features could be good indicators for malignancy of a breast tumor. The Python programming language was used to calculate the Point-Biserial Correlation Coefficients and t-values of each parameter in the University of Wisconsin diagnostic breast cancer dataset (Fig. 8). The Point-Biserial Correlation Coefficient is a value between -1 and 1, inclusive, that measures the strength of association between a continuous variable and a binary variable. The closer the absolute value of the coefficient is to 1, the stronger the association. In this study, 10 different coefficients were calculated, one for each parameter. The feature was the continuous variable and

malignancy was the binary variable. The Point-Biserial Correlation Coefficient, referred to as r_{pb}, is calculated as follows:

$$r_{pb} = \frac{M_1 - M_0}{S_{n-1}} \sqrt{\frac{n_1 n_0}{n(n-1)}} \quad (6)$$

where M1 represents the mean value of a parameter for all malignant tumors, M0 represents the mean value of a parameter for all malignant tumors, n1 is the number of malignant tumors, and n0 is the number of benign tumors. sn-1 is the standard deviation of the parameter for all individuals in the sample:

$$S_{n-1} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \quad (7)$$

A two-tailed t-test was used to test for statistical significance of association between the parameters and malignancy. The null hypothesis is that there is no association between each of the parameters and malignancy. The alternative hypothesis was that there is a correlation between each of the parameters and malignancy. The significance level selected was $p < .05$. Before the t-test can be performed, several conditions must be satisfied. First, we need to check if the sample is random. While it is very likely that the samples from the University of Wisconsin are not random, it would be unethical to randomly select and assign breast cancer to people. Second, there must be independence. This can be checked by either seeing if there is replacement or by checking the 10% condition, which states that independence can be assumed if the sample size is less than 10% of the population size. The dataset from the University of Wisconsin has only 569 tumors, and there are certainly more than 5690 occurrences of tumors, so the 10% condition is satisfied, which means that independence can be assumed. Third, the Normal/Large Condition for sample means must be satisfied. If the population is normal, then the sampling distribution of sample means is also normal. If the population is not normal, the Central Limit Theorem says that the sampling distribution of sample means will be approximately normal in most cases if the sample size is greater than 30. Since the size of the dataset is 569, which is larger than 30, the Normal/Large Condition for sample means is satisfied, so a normal approximation can be used.

Since all three conditions have been met, the t-test can be used to test for statistical significance in the association between each of the ten features and malignancy. The t-value is calculated as follows:

$$t = r_{pb}\sqrt{\frac{n_1 + n_0 - 2}{1 - r_{pb}^2}} \quad (8)$$

n1 and n0 represent the number of malignant and benign tumors, respectively. r_{pb} is the Point-Biserial Correlation Coefficient that was calculated earlier. n1 + n0 - 2 is the degrees of freedom. The t-value calculated for each parameter was then used in the *tcdf()* function on a TI-84 Plus Silver Edition calculator. *tcdf()* takes in three parameters: a lower bound, an upper bound, and degrees of freedom. The p-value for each parameter was calculated:

$$p = 1 - tcdf(-|t|, |t|, 567) \quad (9)$$

The null hypothesis was rejected for a parameter if that parameter's p-value was less than .05. If the null hypothesis was rejected, then there is sufficient statistical evidence that there is an association between that parameter and malignancy.

3 Results

The decision tree model can return a prediction of whether the tumor is malignant or benign, given a row of data in which the ten parameters are radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension, and the decision tree can also output the accuracy of the prediction (Fig. 2). An iOS mobile app (Fig. 5) was developed to provide a user interface to radiologists or other specialists who can input the values of the ten parameters, which are then sent to the machine learning algorithm to make a prediction and calculate the prediction accuracy. These results are sent back to the app, which displays the prediction and the prediction accuracy. Because data points were randomly chosen for the training and testing samples in the *testPredictiveModel()* function, accuracy was not the exact same every time. However, accuracy was consistently between 90% to 93%. Printing the decision tree (Fig. 3) showed that concave points was the group used for the root node, meaning that the concave points group yielded the lowest Gini index of all groups. This could mean that of the ten parameters used, concave points could be the best indicator of whether a breast tumor is malignant or benign. When max_depth is set to be 5, not all the parameters are used for the model. The root node uses concave points, and the decision nodes use a variety of parameters: texture, area, concave points, radius, smoothness, perimeter, and concavity. However, compactness, symmetry, and fractal dimension were not included in the decision tree at all.

The t-test, using the Point-Biserial Correlation Coefficient, showed which parameters had an association with malignancy (Fig. 4). If $p < .05$, then the null hypothesis is rejected, showing convincing statistical evidence that there is an association between the parameter and malignancy. Radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, and symmetry had $p < .05$. Fractal dimension, on the other hand, did not accepted the null hypothesis. The features that had a relatively high Point-Biserial Correlation Coefficient

were concave points (r_{pb} = 0.777), perimeter (r_{pb} = .743), radius (r_{pb} = .73), area (r_{pb} = .709), and concavity (r_{pb} = .696).

4 Discussion

This decision tree model can be used to detect breast cancer in patients. Since this is only a prediction, this should not be the only means of determining if a breast tumor is malignant or benign; however, it can give doctors a good sense of the severity of the tumor is harmful or not. If used at an early stage, the diagnosis from this model can help doctors take immediate action to address any possible malignant tumors. As there are 1.5 million women affected by breast cancer every year, this technology can help thousands of women get early treatment.

While the project was geared toward breast cancer, this algorithm can be applied to just about any problem which requires binary classification. If a patient has a type of cancer that is not breast cancer, such as skin cancer, but the doctor has the ten parameters used in this project, they could use a similar program and would get results with very similar accuracy. Other scenarios of binary classification could use this model as well; the only parts of the code that would have to change are the *processRawData()* function, the name of the text file that has the data, and the lines of code that display the results. This algorithm provides an abstraction in the form of functions that a user could use to develop a machine learning program for binary classification.

A possible next step is to add a random forest algorithm in addition to the CART algorithm. While CART uses only one decision tree, random forest uses multiple decision trees. In random forest, each of the trees will return a prediction of the class value. Whichever class value has the highest frequency will be the value that the program returns to the user. Random forest is incredibly efficient because it can handle data with many attributes and it can use dimensional reduction methods to remove variables that do not seem to have a strong association with the class labels. The bar charts in Fig. 1 display the data for each feature. Dimensional reduction could be used to eliminate variables that do not show a strong association with the class values. For example, as seen in Fig. 1, malignant and benign tumors can both have low, medium or high fractal dimensions, so fractal dimension would not help predict the malignancy of a breast tumor. By forming multiple decision trees and limiting the features used in the model, random forest can improve the accuracy of the prediction algorithm.

Another possible next step would be to pre-prune parameters that either show no association or a very weak association with malignancy. When performing the t-test, it was found that fractal dimension did not achieve $p < .05$, so the null hypothesis that there is no association is accepted. In addition to this, the $|r_{pb}|$ for fractal dimension was .013, again showing that there is a lack of association between fractal dimension and malignancy. While texture (r_{pb} = .415), smoothness (r_{pb} = .359), and symmetry (r_{pb} = .33) all showed association with malignancy through the t-test, their r_{pb} values are relatively low, so those associations may not be very strong. So, for modification of the current program, fractal dimension, symmetry, smoothness, and texture could all be pruned before execution of the CART algorithm. This would result in a decision tree that is solely based on parameters that show a relatively strong association with malignancy, potentially increasing prediction accuracy.

5 Appendix

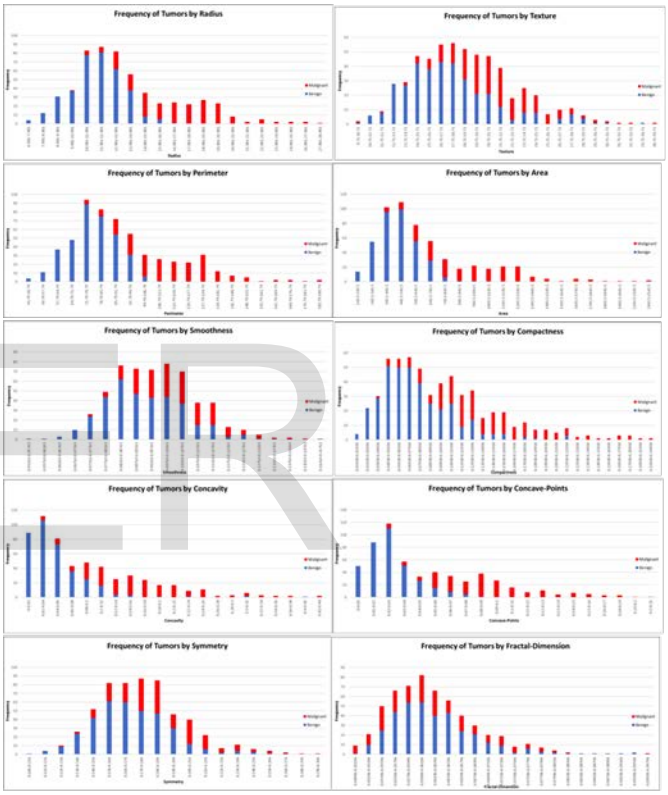

Fig. 1. Frequency of tumors by attribute (radius, texture, perimeter, area, smoothness, compactness, concavity, concavity points, symmetry, and fractal dimension). Red bars represent malignant tumors, and blue bars represent benign tumors.

```
Breast-Cancer-Prediction-Algorithm — python breast_canc...
************************************************************
*              Breast Cancer Diagnosis Program             *
*                    by Aniruddha Murali                   *
************************************************************
Loading Breast Cancer Data: Successful
Testing model...
Model Test: Passed
Model Prediction Accuracy (Mean): 92.416%


Please input patient information:
        Enter Radius: 17.99
        Enter Texture: 10.38
        Enter Perimeter: 122.8
        Enter Area: 1001
        Enter Smoothness: 0.1184
        Enter Compactness: 0.2776
        Enter Concavity: 0.3001
        Enter Concavity Points: 0.1471
        Enter Symmetry: 0.2419
        Enter Fractal Dimension: 0.07871

Processing patient data...

Tumor Diagnosis Results:
-----------------------
Prediction: MALIGNANT
Prediction Accuracy (Mean): 92.416%
```

Fig. 4. Statistical analysis using the Point-Biserial Correlation Coefficient and a t-test.

Fig. 2. Output of machine learning program in Command Line

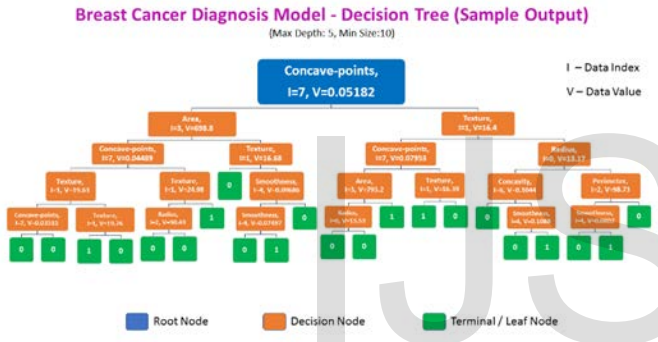

Fig. 3. Example of decision tree model with max_depth = 5, min_size = 10 (tree has a height of 5 with a minimum of 10 training patterns at each decision node).

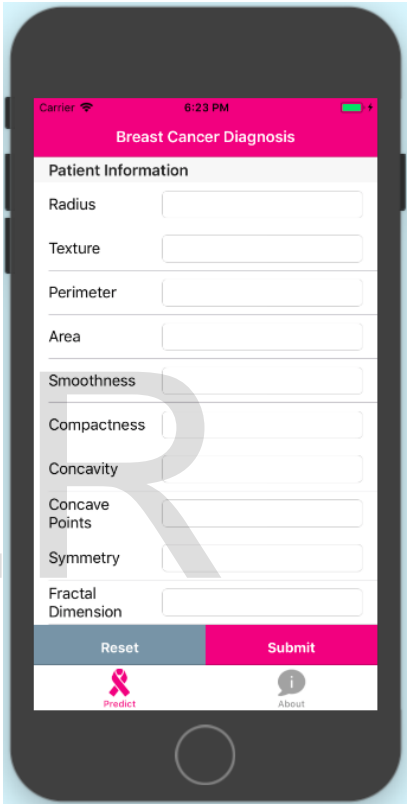

Fig. 5. iOS mobile app allows a user to input the values of the 10 features of a breast tumor.

Point-Biserial Correlation Test Results

Parameter	r_{pb}	Degrees of Freedom	t-value	p-value	p < .05
Radius	0.73	567	25.436	~ 0	Yes
Texture	0.415	567	10.867	~ 0	Yes
Perimeter	0.743	567	26.405	~ 0	Yes
Area	0.709	567	23.939	~ 0	Yes
Smoothness	0.359	567	9.146	~ 0	Yes
Compactness	0.597	567	17.698	~ 0	Yes
Concavity	0.696	567	23.104	~ 0	Yes
Concave-Points	0.777	567	29.354	~ 0	Yes
Symmetry	0.33	567	8.338	~ 0	Yes
Fractal Dimension	-0.013	567	-0.306	0.76	No

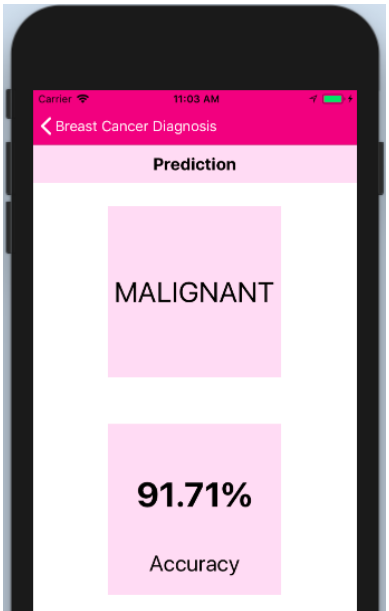

Fig. 6. iOS mobile app returns prediction of malignancy and the accuracy of the prediction.

```
''' calculateGiniIndex()
* This method calculates the Gini index for a split dataset
*
* @param groups - features of the object
* @param classes - The possible labels of the groups
'''
def calculateGiniIndex(groups, classes):
	# count all samples at split point
	total_samples = float(sum([len(group) for group in groups]))
	# sum weighted Gini index for each group
	gini = 0.0
	for group in groups:
		size = float(len(group))
		# avoid dividing by zero
		if size == 0:
			continue
		score = 0.0
		# score the group based on the score for each class
		for class_val in classes:
			p = [row[-1] for row in group].count(class_val) / size
			score += p * p
		# weigh the group score by its relative size
		gini += (1.0 - score) * (size / total_samples)
	return gini

''' getSplitBestNode()
* This method selects the best split point for a dataset
*
* @param dataset - Data set that is inputted to find best split point
'''
def getSplitBestNode(dataset):
	class_values = list(set(row[-1] for row in dataset))
	best_index, best_value, best_score, best_groups = 999, 999, 999, None
	for index in range(len(dataset[0])-1):
		for row in dataset:
			groups = splitGroup(index, row[index], dataset)
			gini = calculateGiniIndex(groups, class_values)
			if gini < best_score:
				best_index, best_value, best_score, best_groups = index,
				row[index], gini, groups
	return {'index':best_index, 'value':best_value, 'groups':best_groups}

''' splitNode()
* This method creates child splits for a node or creates a terminal
*
* @param node - decision node of decision tree that is to be split into sub-nodes
* @param max_depth - The maximum height of the decision tree
* @param min_size - The minimum size of a decision tree
* @param depth - Current depth of tree
'''
def splitNode(node, max_depth, min_size, depth):
	left, right = node['groups']
	del(node['groups'])
	# check for no split
	if not left or not right:
		node['left'] = node['right'] = toTerminalNode(left + right)
		return
	# check for max depth
	if depth >= max_depth:
		node['left'], node['right'] = toTerminalNode(left), toTerminalNode(right)
		return
	# process left group
	if len(left) <= min_size:
		node['left'] = toTerminalNode(left)
	else:
		node['left'] = getSplitBestNode(left)
		splitNode(node['left'], max_depth, min_size, depth+1)
	# process right group
```

```
# Evaluate an algorithm using a ten-fold cross validation split
''' testPredictiveModel()
* This method splits the data into sets and returns the mean accuracy of the model
*
* @param train - training data
* @param test - testing data
* @param max_depth - Maximum depth of the tree
* @param min_size - minimum size of the tree
'''
def testPredictiveModel(dataset, random_set_splits, max_depth, min_size):
	scores = list()
	sets = splitData(dataset, random_set_splits)

	for random_set in sets:
		train_set = list(sets)
		train_set.remove(random_set)
		train_set = sum(train_set, [])
		test_set = list()
		for row in random_set:
			row_copy = list(row)
			test_set.append(row_copy)
			row_copy[-1] = None

		predictions = testPredictions(train_set, test_set, max_depth, min_size)
		actual = [row[-1] for row in random_set]
		accuracy = calculateAccuracy(actual, predictions)
		scores.append(accuracy)

	return (sum(scores)/float(len(scores)))
```

Fig. 7. Sample of functions used in machine learning algorithm to build predictive model.

```
'''
Note: All arrays printed show calculations for the parameters in the following order:
Radius, Texture, Perimeter, Area, Smoothness, Compactness, Concavity, Concave points,
Symmetry, Fractal Dimension
'''

import math

'''Returns values of variables used in other calculations'''
def setup():
    name = "breast_cancer_data.txt"
    file = open(name, 'r')
    count = 0
    benignCount = 0
    malignantCount = 0
    tumors = list()
    benignMeans = list(0 for i in range(0,10))
    malignantMeans = list(0 for i in range(0,10))
    means = list(0 for i in range(0,10))

    for line in file:
        line = line.split(',')
        tumor = [float(line[2]), float(line[3]), float(line[4]), float(line[5]),
                 float(line[6]), float(line[7]), float(line[8]), float(line[9]),
                 float(line[10]), float(line[11]), line[1]]
        tumors.append(tumor)
        count += 1

    for tumor in tumors:
        if tumor[len(tumor)-1] == "M":
            malignantCount += 1
            for i in range(0,10):
                malignantMeans[i] += tumor[i]
        elif tumor[len(tumor)-1] == "B":
            benignCount += 1
            for j in range(0,10):
                benignMeans[j] += tumor[j]
        for k in range(0,10):
            means[k] += tumor[k]

    for b in range(0,len(benignMeans)):
        benignMeans[b] /= benignCount
```

```
'''Calculates standard deviation of a dataset'''
def stdev(nums, average):
    diff = 0
    for n in nums:
        diff += (n - average)**2
    return math.sqrt(diff/(len(nums)-1))

'''Calculates Point-Biserial Correlation Coefficient of a parameter'''
def rpb():
    n0, n1, bm, mm, meanList, tumorList = setup()
    n = n0 + n1
    rpbs = list()

    for i in range(0, len(bm)):
        m1 = mm[i]
        m0 = bm[i]
        mean = meanList[i]

        paramList = list(tumorList[j][i] for j in range(0,n))
        sd = stdev(paramList, mean)

        rpb = ((m1-m0)/sd) * math.sqrt((n1*n0)/(n*(n-1)))
        rpbs.append(rpb)

    print("rpb: ")
    print(rpbs)
    print(' ')
    return rpbs, n0, n1

'''Calculates the t-value of a parameter'''
def t():
    rpbList, n0, n1 = rpb()
    ts = list()
    for r in rpbList:
        dof = n0 + n1 - 2
        denom = 1 - r**2
        t = r * math.sqrt(dof/denom)
        ts.append(t)

    print("t-values: ")
    print(ts)

print("\nStatistical Analysis of University of Wisconsin Diagnostic Breast Cancer Dataset:")
print("--------------------------------------------------------------------------------\n")
print("Note: All arrays printed show calculations for the parameters in the following order:
Radius, Texture, Perimeter, Area, Smoothness, Compactness, Concavity, Concave points,
Symmetry, Fractal Dimension\n")

t()
```

Fig. 8. Python code used to calculate Point-Biserial Correlation Coefficients and t-values.

ACKNOWLEDGMENT

The author would like to thank Mrs. Karen Thompson for mentoring him on the overall research process and Mr. Phillip Abraham for mentoring him on statistical analysis.

REFERENCES

[1] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Wolberg, W., Street, N., & Mangasarian, O. (1995, May). Image analysis and machine learning applied to breast cancer diagnosis and prognosis. International Academy of Cytology.

[3] Early Diagnosis and Screening - Breast Cancer. (n.d.). Retrieved from World Health Organization website: http://www.who.int/cancer/prevention/diagnosis-screening/breast-cancer/en/

[4] Vnencak-Jones, C., M. Berger, W. Pao. 2016. Types of Molecular Tumor Testing. My Cancer Genome https://www.mycancergenome.org/content/molecular-medicine/types-of-molecular-tumor-testing/

[5] Song, Y.-Y., & Lu, Y. (n.d.). Decision tree methods: applications for classification and prediction. Retrieved from US National Library of Medicine National Institutes of Health website: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/#B14

[6] A Complete Tutorial on Tree Based Modeling from Scratch. (n.d.). Retrieved from Analytics Vidhya website: https://www.analyticsvidhya.com/blog/2016/04/complete-tutorial-tree-based-modeling-scratch-in-python/

[7] Mammography. (n.d.). Retrieved from National Institute of Biomedical Imaging and Bioengineering website: https://www.nibib.nih.gov/science-education/science-topics/mammography

[8] Devitt, J. E. (1985, January 31). Screening for breast cancer: current status, problems, prospects. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2327346/

[9] Kingsford, C., & Salzberg, S. L. (2008, September 26). What are decision trees? Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2701298/

[10] Decision Trees. (n.d.). Retrieved from Scikit Learn website: http://scikit-learn.org/stable/modules/tree.html#classification

[11] Conduct and Interpret a Point-Biserial Correlation. (n.d.). Retrieved January 26, 2018, from Statistics Solutions website: http://www.statisticssolutions.com/point-biserial-correlation/

[12] Point-biserial correlation coefficient. (n.d.). Retrieved January 26, 2018, from Wikipedia website: https://en.wikipedia.org/wiki/Point-biserial_correlation_coefficient