Identifying Breast Cancer

CSCI-3351

**Bytecode Boys**

****

University of New Haven

TAGLIATELA COLLEGE OF ENGINEERING, West Haven, CT

Submitted To:

Prof Reza Sadeghi

Fall 2020

Table of Contents

[**Project Report: Phase – 6**](#_p82gbgs4urdz) **3**

[Team Information](#_t3j6gke1kf5k) 3

[Team Name](#_8od5tnw2a4k8) 3

[Team Members](#_vr8n0a87rh0) 3

[Research Question](#_n69dh4esf1ox) 3

[Research Objective](#_blurleyj353o) 3

[Literature Review](#_3jyr8koy4zaq) 3

[About the Dataset](#_5gi4rjjyml8r) 4

[Data Collection Methods](#_n8m0v2vg7jad) 4

[Accuracy of the Dataset](#_33ytv6rodoc6) 4

[Plan for Solving](#_3qdugdwnwfw5) 4

[Results](#_5tagnbrvp76t) 5

[Decision Tree (Bobby Chenkus)](#_57knjosg5ilo) 5

[MLP Classifier (Jake Intravaia)](#_qxrdvpv76q78) 8

[Random Forest Tree (Trey Gary)](#_wmkon8rqbf60) 10

[References](#_c53tw8stlelm) 13

# Project Report: Phase – 6

## Team Information

### Team Name

Bytecode Boys

### Team Members

1. Jake Intravaia : [jintr1@unh.newhaven.edu](mailto:jintr1@unh.newhaven.edu) (Team Head)
2. Bobby Chenkus : [rchen5@unh.newhaven.edu](mailto:rchen5@unh.newhaven.edu) (Team Member)
3. Trey Gary : [tgary1@unh.newhaven.edu](mailto:tgary1@unh.newhaven.edu) (Team Member)
4. Alexander Russello : [aruss10@unh.newhaven.edu](mailto:aruss10@unh.newhaven.edu) (Team Member)
5. Nathaniel Small : [nsmal1@unh.newhaven.edu](mailto:nsmal1@unh.newhaven.edu) (Team Member)

## Research Question

Can we use machine learning and patient data to accurately identify malignant breast cancer in patients by analyzing previously diagnosed patients?

## Research Objective

The main objective of the project is to create a Neural Network capable of accurately predicting the presence of malignant breast cancer in patients using patient data.

## Literature Review

It is known that detection, prevention, and treatments for different cancers are being developed. However, breast cancer is one of the most common cancers among women and the second most common cancer overall [1][2].

Breast cancer research opens the door to finding better ways to prevent, detect, and treat breast cancer. This project’s analysis aims to observe which attributes are most helpful in predicting a patient’s degree of malignance. General trends will also be observed to enable the selection of attributes that will yield the highest percentage of correctly guessed cases. The goal is to classify the degree of a patient’s malignance. Machine learning classification methods were used to fit a function that can, to an extent, predict degree of malignancy based upon several different inputs.

## About the Dataset

The dataset covers the data from breast cancer cases reported by the Oncology Institute. This set contains a total of 286 instances for analysis which is split into two separate classes. There are a total of ten attributes which includes class, age, menopause, tumor-size, inv-nodes, node-caps, deg-malig, breast, breast-quad, and irradiant.

It’s also important to note that this dataset is from 1988. This dataset has been referenced many times, but was last referenced in 2005 [3]. For the purpose of this project this dataset is useful, however, a newer dataset should be used for an thorough and updated analysis of this issue.

### Data Collection Methods

The data was collected by Matjaz Zwitter & Milan Soklic from the Oncology Institute. This institute is known for having datasets used for machine learning applications.

### Accuracy of the Dataset

The dataset has been provided by the Oncology Institute and is published by the UCI machine learning repository. This repository is supported by the Nation Science Foundation (NSF) which supports the fundamental research and education in all of science and engineering. The process of machine learning can be classified as computer science which the NSF is the major source of federal backing. With this information, the dataset provided can be considered accurate.

### Plan for Solving

Initially our dataset was a mixture of strings and numerical values, separated by commas which was not ideal for our neural network. Our team leader created a convert.py file that allowed us to convert our mixed data into all numerical values for use within our neural network models. This convert.py file and the outputted numerical value would be the base data in which we all based our models off of.

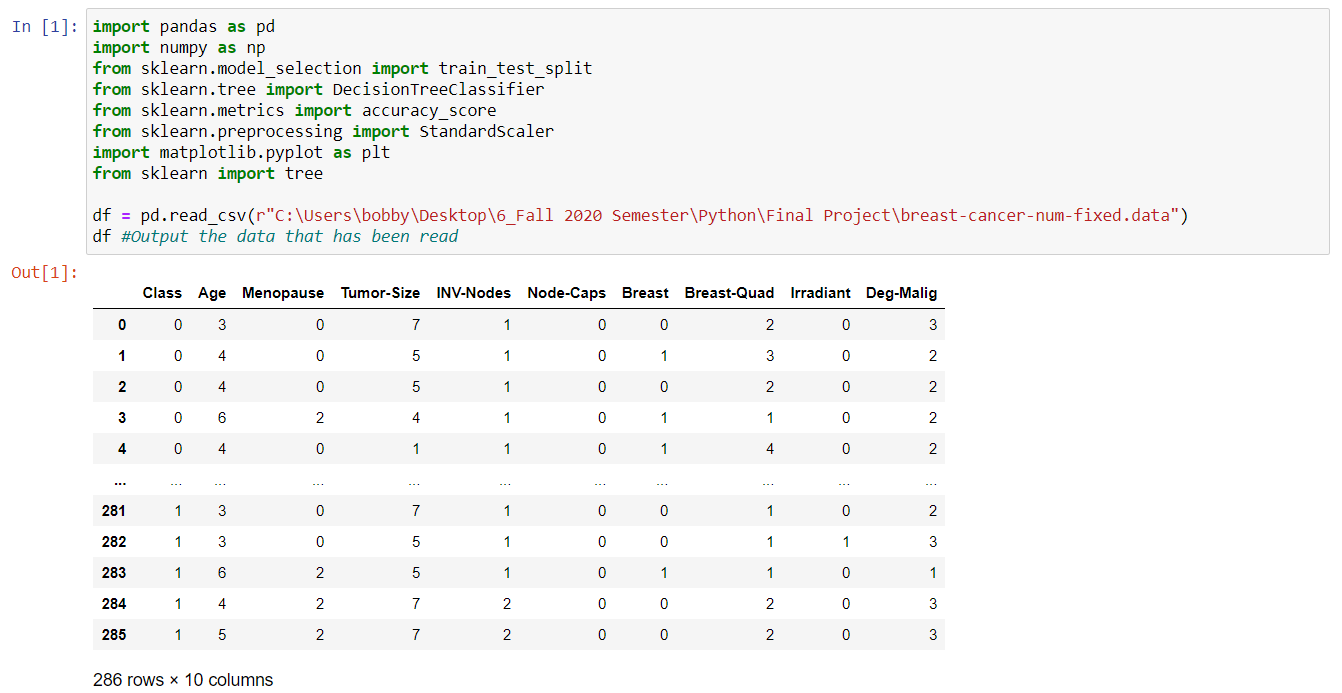
Once we had our numerical data, we could begin to create our different Neural Network models, utilizing a variety of modules from scikit learn to try and get our predictions to be as accurate as possible.

## Results

### Decision Tree (**Bobby Chenkus**)

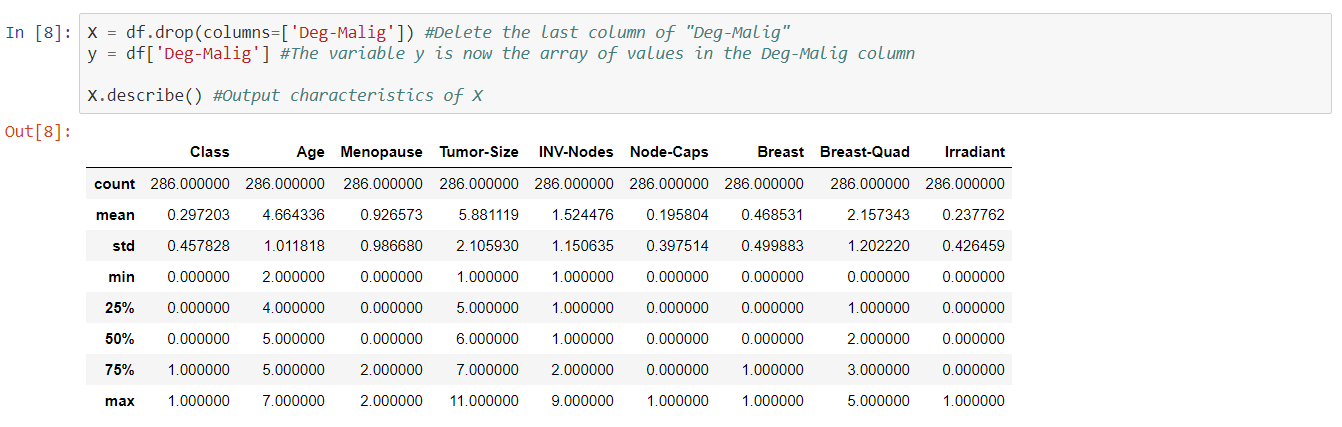
Scikit-Learn’s decision tree module seemed to be a good fit for this dataset due to its ability to look for trends not only within numbers, but also within categorical data. This versatile method of classification sounds like a good method to analyze many different kinds of datasets. The visualization aspect of this classification is also appealing so the decisions through each step of the “tree” can be seen.

All libraries were first imported. The variable *df* was declared and used to read the converted .py file containing the dataset’s information. To ensure that the file was read properly df was output to the terminal showing the full dataset.

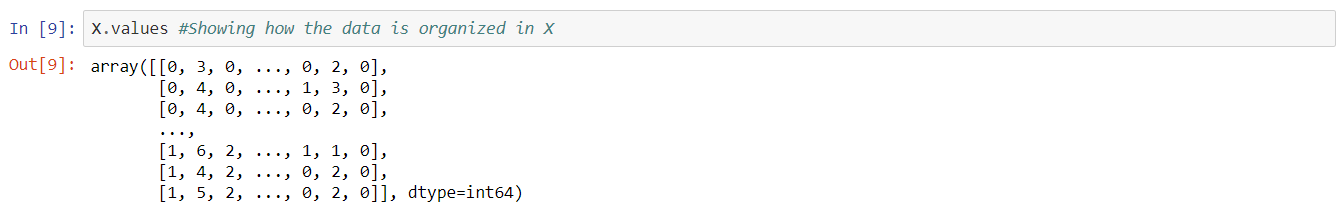


**Figure 1.** Imported libraries and complete dataset

Variables X and y were declared to separate the dataset into two separate arrays of information. X.describe() was used to show some of the characteristics of the array and allowed for confirmation that Deg-Malig was dropped from the attributes. The values within variable X were output to show how the data in the array was organized.

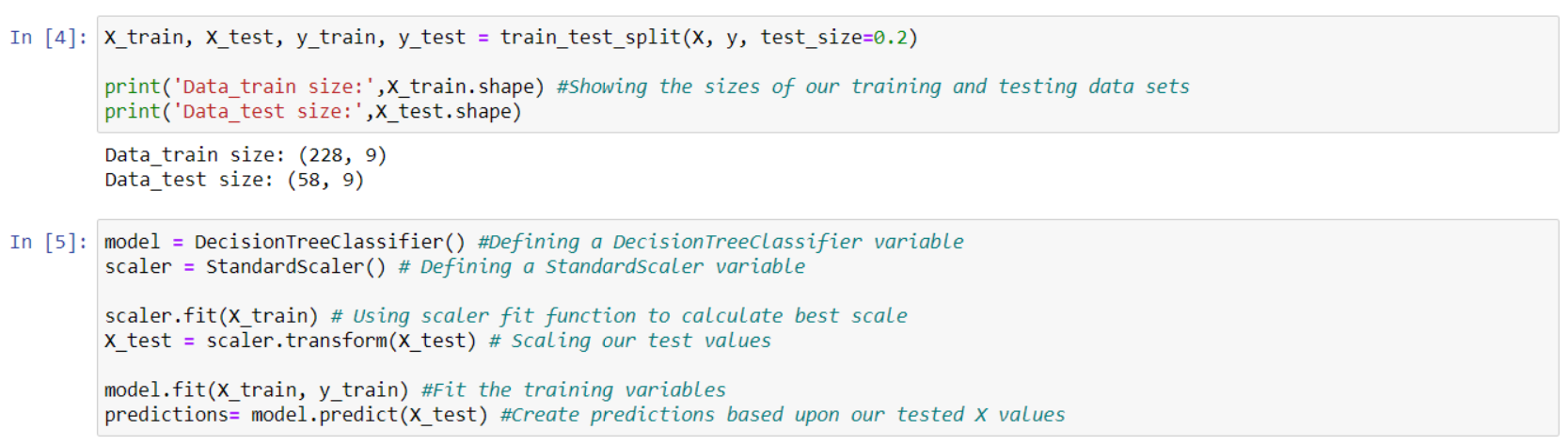


**Figure 2.** Characteristics of dataset X



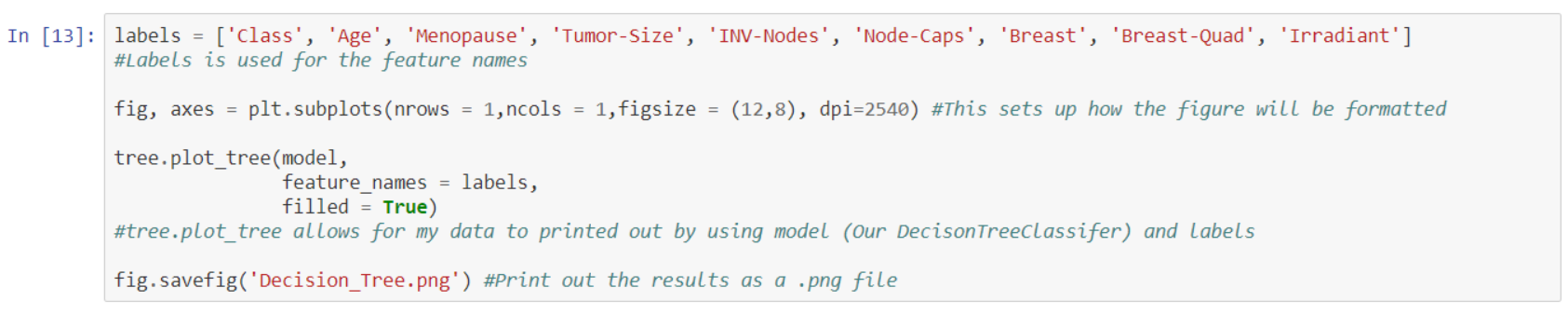
**Figure 3.** Values within the X dataset

The training and testing then began by using train\_test\_split from sklearn.model. The sizes of the training and testing datasets were printed to ensure that the splitting of data was done properly. Model was declared as a variable for the DecisionTreeClassifier() which would allow the training variables to be fit into the decision tree classification system. However, the training and testing data was first fit to a scalar function to create a number scaling format which would allow for properly scaled results.

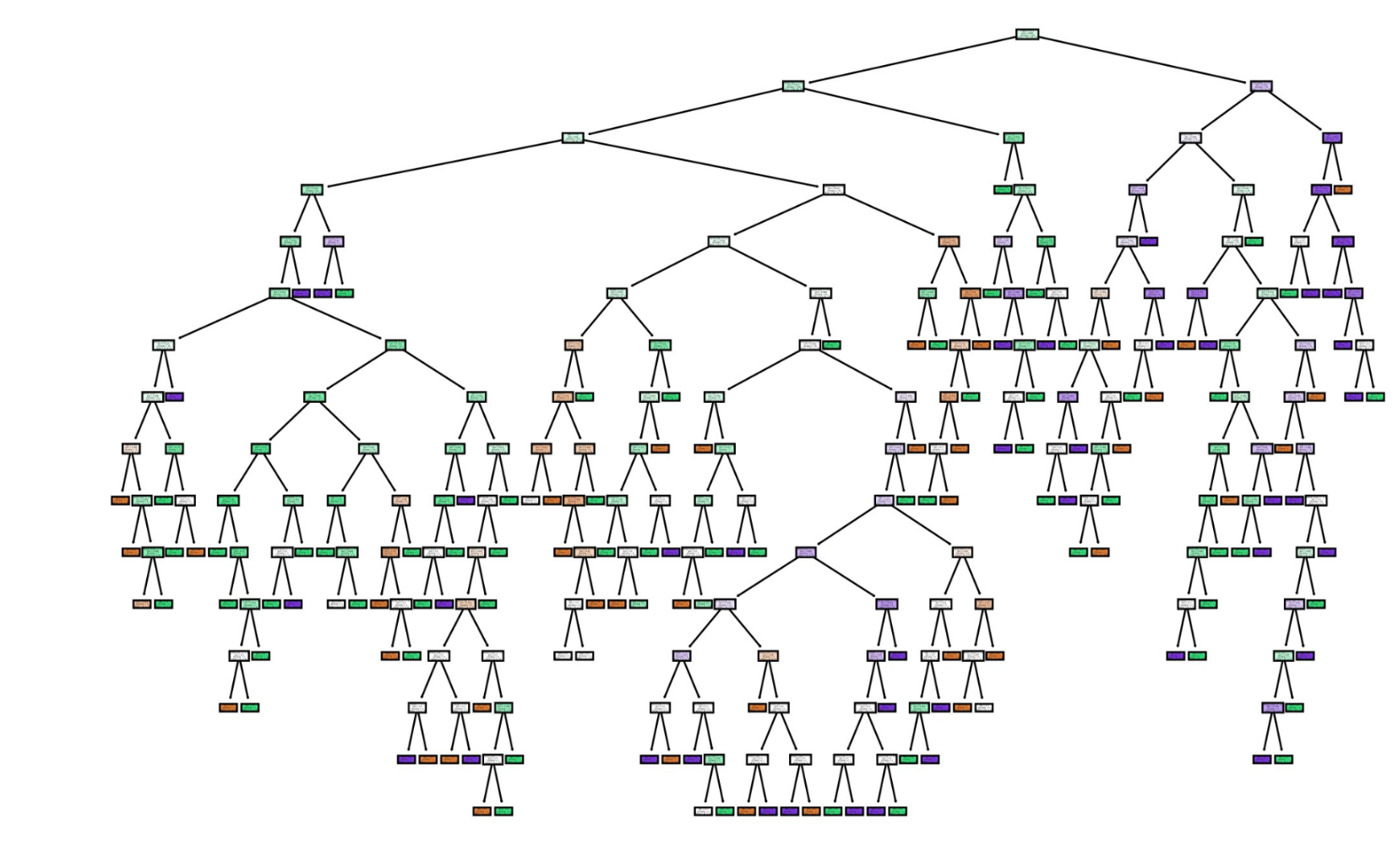


**Figure 4.** Train and Test Data Split & Fitting to Standard Scale and Decision Tree

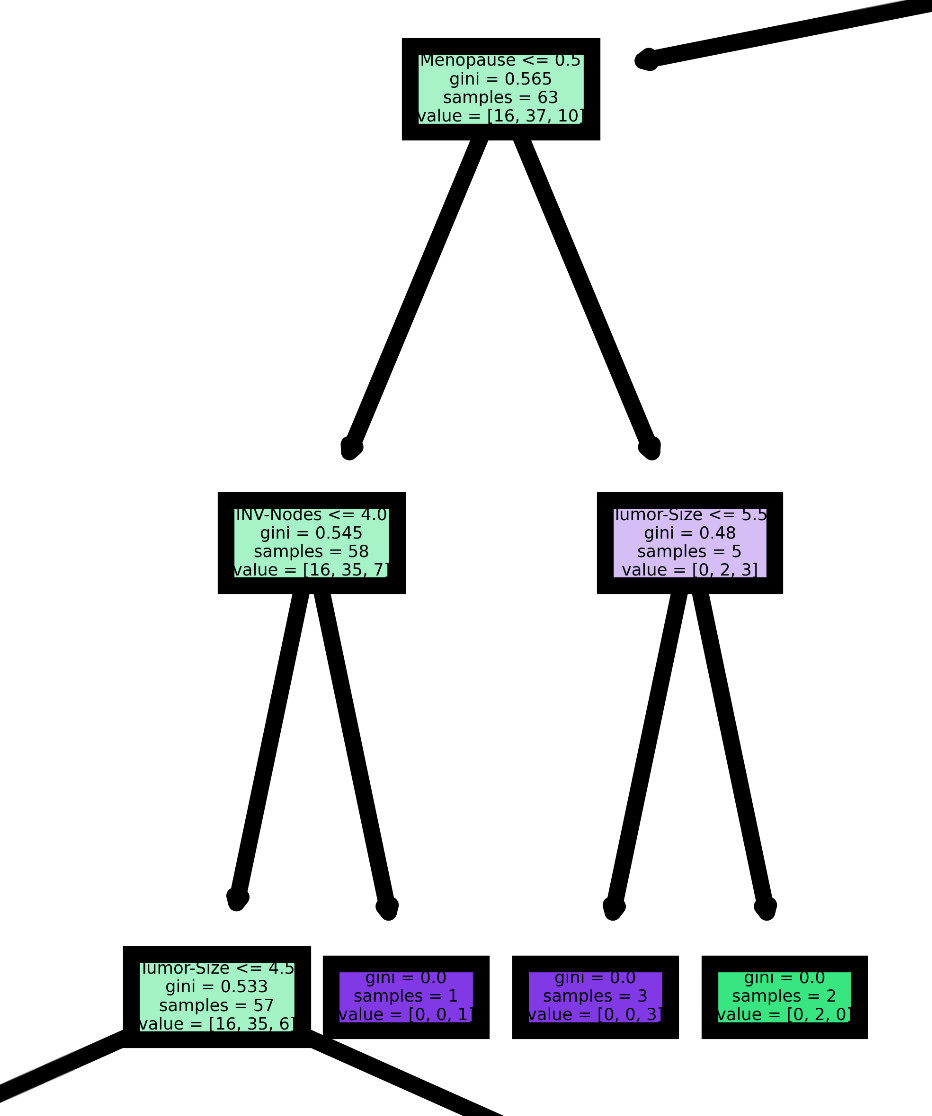
The decision tree was visualized by using the below functions. The labels variable allowed the created tree to create titles for each box and make decisions based on the values of these labels or attributes. Since the scale has been standardized number comparison has been best fit for the numbers within this dataset.



**Figure 5.** Code for Decision Tree Visualization

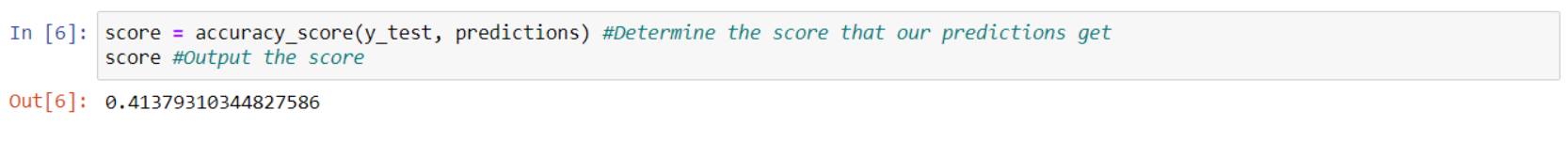


**Figure 6.** Decision Tree Visualization



**Figure 7.** Zoomed in Image of Decisions within Tree

As shown in Figure 8, this method used the accuracy score to show how accurate the predictions made from the training data was when compared to the testing set. The average accuracy score of the decision tree was around 35 - 45%. These numbers seemed to be slightly higher on average when compared to other observed modules. The decision tree worked okay for the analysis of this dataset, however it is not reliable in making consistently accurate predictions. A different analysis should be looked at to determine whether the degree of malignancy can be determined from this dataset’s attributes.

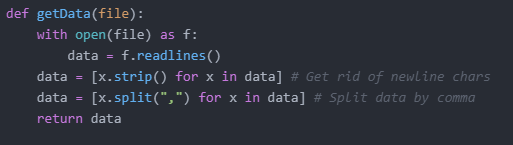


**Figure 8.** Accuracy Score of Decision Tree

### MLP Classifier (**Jake Intravaia**)

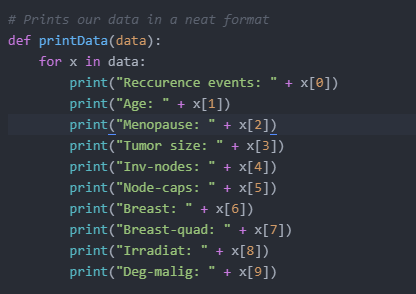
Scikit-Learn’s MLPClassifier module was perfect for the kind of data we had to process. The classifier takes in several attributes and data, and based on that data outputs a certain classification. This works very well for our dataset considering we need to classify the degree of malignance a tumor has based on several attributes of the tumor.

To begin, I imported the data from the text file into python lists that numpy can utilize.



**Figure 9.** Reading Content of the File

Then, just to ensure the integrity of the data, I created a function that printed our data in a human-readable format.



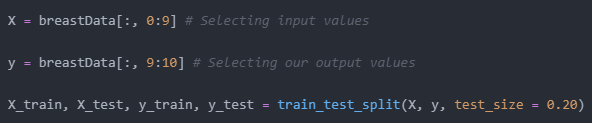
**Figure 10.** Printing Data to Verify Integrity

After ensuring our datas integrity and that it has been imported correctly into lists, I began to initialize our variables by first creating a python list, then converting it to a numpy array.



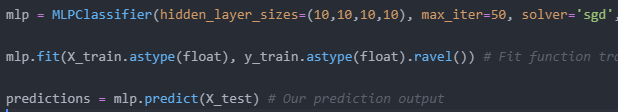
**Figure 11.** Converting Data to a Numpy Array

Then it was time to import our input data (**X**) and the value we are trying to predict (**y**). Once imported, I then split our data into test data and training data. To ensure our values are scaled for use in the MLPClassifier module, I used scikit-learns built in scaler function, that scaled our data into usable numerical data.



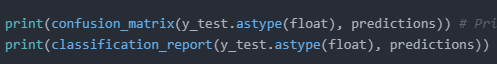
**Figure 12.** MLP Classifier Train and Test Data Split

After initializing all our data and creating a training data set and a test data set, it was time to create our MLPClassifier and fit our data.



**Figure 13.** Fitting Data Using MLPClassifier

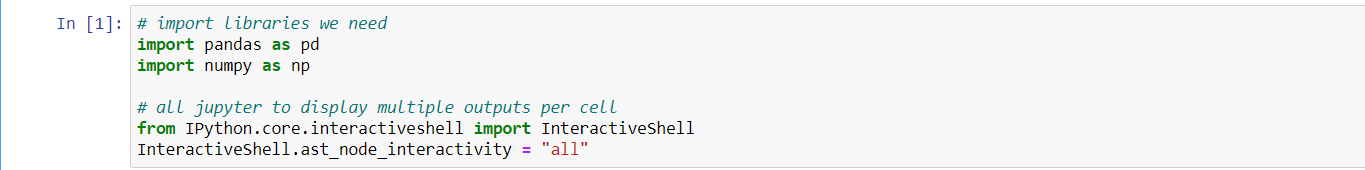
All that was left to do was to print our confusion matrix and classification report to see how accurate our models predictions were.



**Figure 14.** Printing Accuracy of MLP Classifier

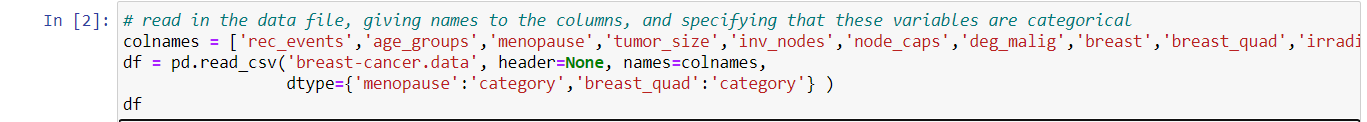
### Random Forest Tree (**Trey Gary & Nathaniel Small**)

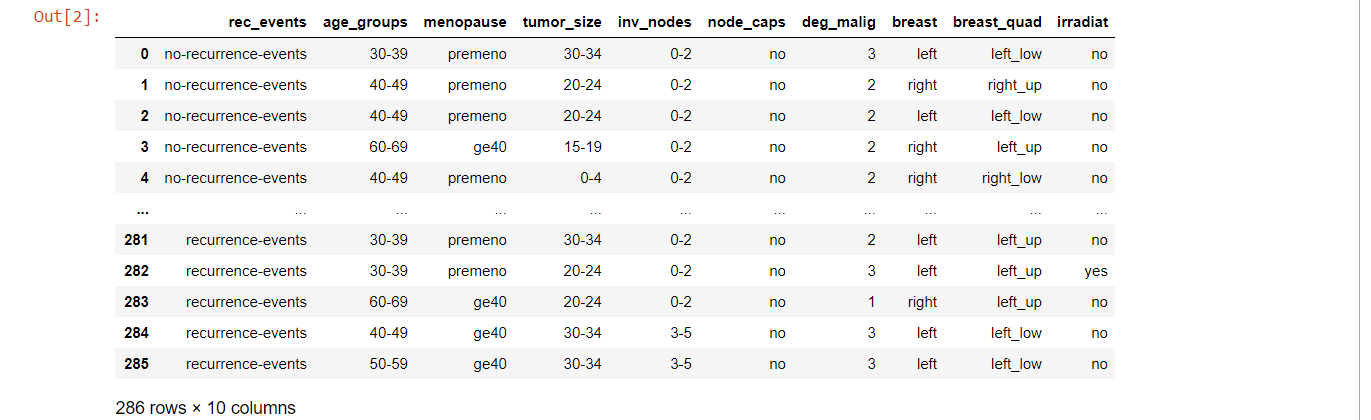
Imported some libraries here as a start.



**Figure 15.** Imported libraries

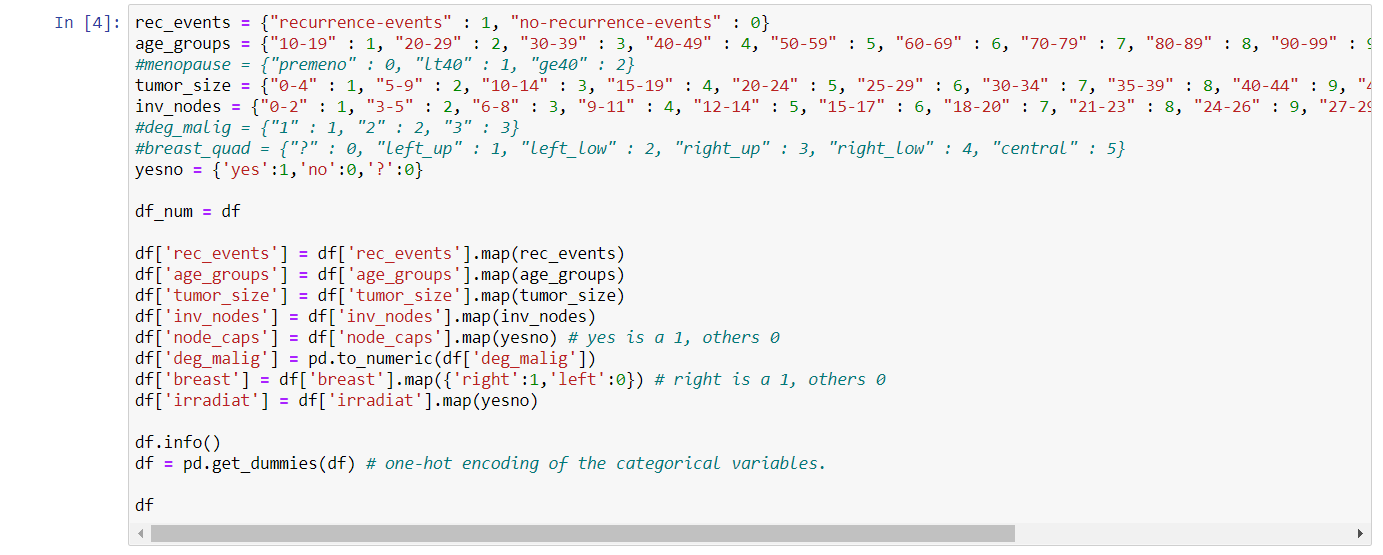
Set the variable df to read in the data file and organize the specified variables through a categorical means. Output also shown below.

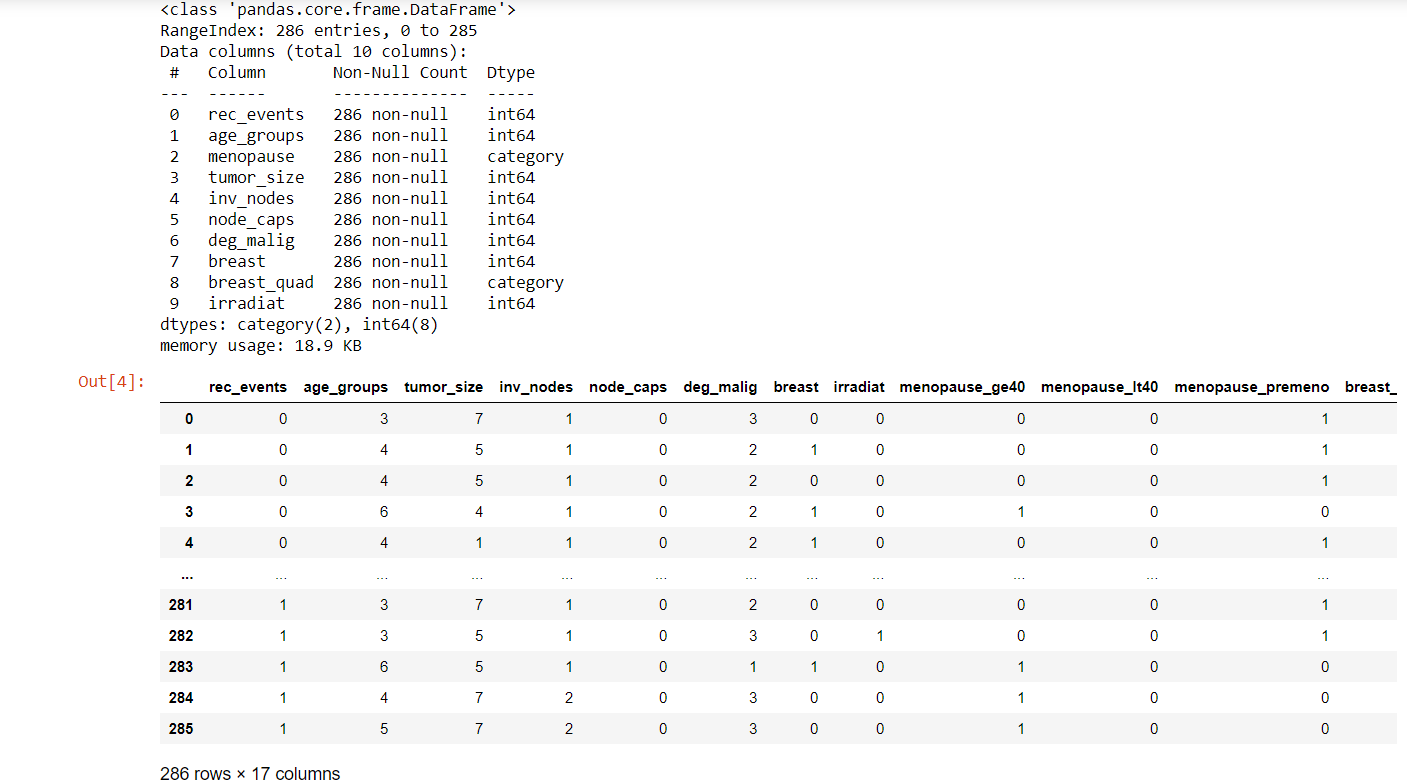




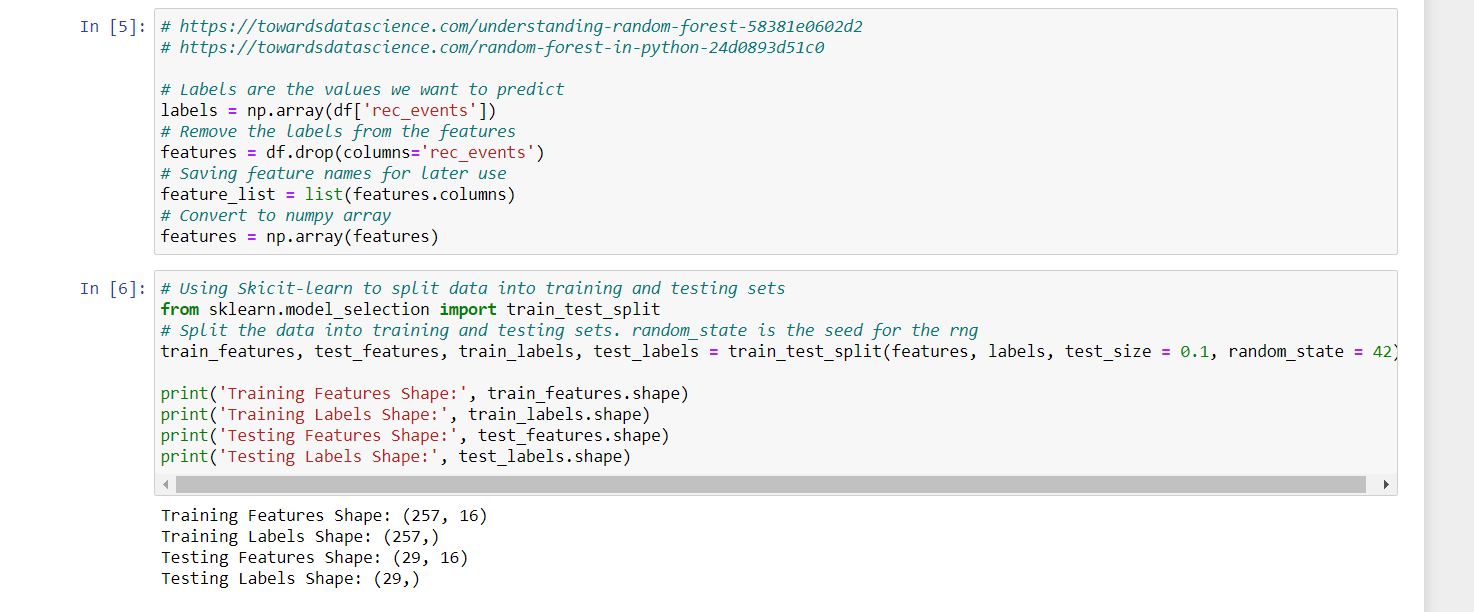
**Figure 16.** Reading in the data file and labeling w/ complete data set

Here is where the data was mapped together into columns for a better visualization and a simple hot encoding method. Also contains a categorized output.



**Figure 17.** Categorizing the data w/output

Imported more libraries such as sklearn and train. Similarly to above, rhe training and testing then began by using train\_test\_split from sklearn.model. The sizes of the training and testing datasets were printed to ensure that the splitting of data was done properly.



**Figure 18.** Train and Test Data Split with labels

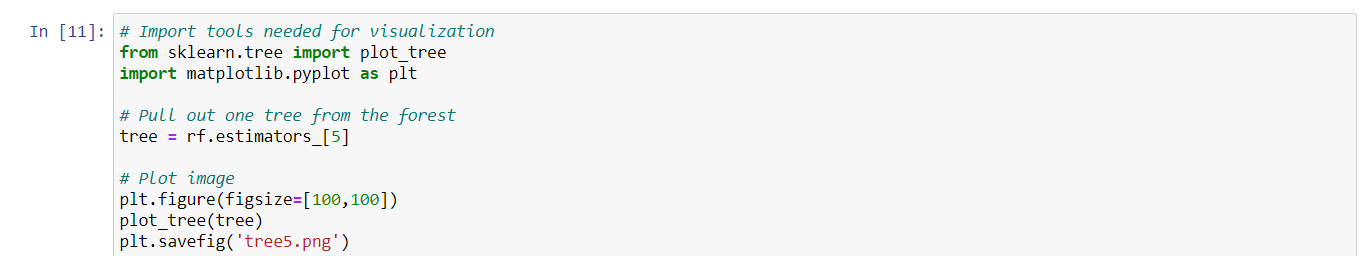
Mean Absolute Error for Random Forest Model: 0.356

Accuracy computed using geometric mean on test data: 55.312%



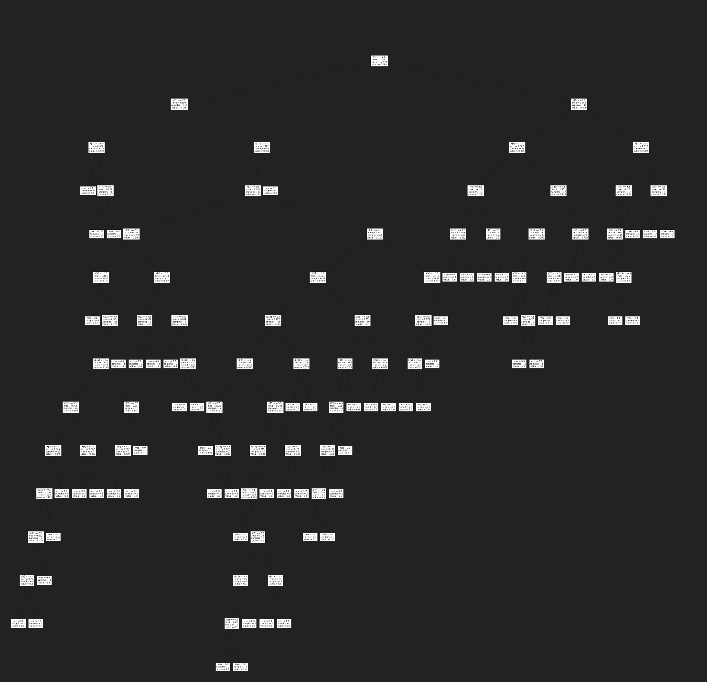
**Figure 19.** Accuracy scores

Random forest was configured and displayed using more imported libraries, and functions that pulled one tree from the forest.



**Figure 20.** Code for display Random Forest

Almost similar representation of tree diagram



**Figure 21.** Random Forest Image

## 

## References

1. <https://www.wcrf.org/dietandcancer/cancer-trends/breast-cancer-statistics>
2. <https://www.bcrf.org/breast-cancer-statistics-and-resources>
3. <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer>
4. <https://towardsdatascience.com/understanding-random-forest-58381e0602d2>
5. <https://towardsdatascience.com/random-forest-in-python-24d0893d51c0>

## GitHub

1. https://github.com/tgary1/Random-Forest
2. https://github.com/BobbyChenkus
3. https://github.com/jakeintravaia/breast-cancer-classifier