CS 412
Team D: Tanuj Dave, Nikith Rachakonda , Aditya Pandey
May 5th 2022
Final Project Report

Breast Cancer Tendency Prediction

Introduction

We chose the Breast Cancer Wisconsin data set by Dr. William H Wolberg at University of Wisconsin Hospitals taken during his studies from 1989 to 1991. The dataset contains data of 699 patients containing various cancerous cell information like thickness, cell size, shape, nuclei information and its tendency – Benign (not aggressive/harmful) or Malignant (aggressive/harmful). We chose two datasets developed from the original study:

1. Breast Cancer Wisconsin (Original)
2. Breast Cancer Wisconsin (Diagnostic)

Datasets

The Breast Cancer Wisconsin (Original) dataset is a multivariate dataset containing 699 entries with 11 attributes. The attributes are all integer values and are as follows:

1. Sample code number: id number
2. Clump Thickness (integer): 1 - 10
3. Uniformity of Cell Size (integer): 1 - 10
4. Uniformity of Cell Shape (integer): 1 - 10
5. Marginal Adhesion (integer): 1 - 10
6. Single Epithelial Cell Size (integer): 1 - 10
7. Bare Nuclei (integer): 1 - 10
8. Bland Chromatin (integer): 1 - 10
9. Normal Nucleoli (integer): 1 - 10
10. Mitoses (integer): 1 - 10
11. Class: (2 for benign, 4 for malignant)

The Breast Cancer Wisconsin (Diagnostic) dataset is also a multivariate dataset. But, it contains features of the breast mass in the original study computed using fine needle aspirates of the digitized image, they also describe characteristics of the cell nuclei present in the images. The features are real rather than integers, and have excluded some of the entries due to missing images/data and thus contain 569 entries rather than 699. The attribute information is included below:

1) ID number
2) Diagnosis (M = malignant, B = benign)

3-32) Ten real-valued features are computed for each cell nucleus:
   a) radius (mean of distances from center to points on the perimeter)
   b) texture (standard deviation of gray-scale values)
   c) perimeter
   d) area
   e) smoothness (local variation in radius lengths)
   f) compactness (perimeter^2 / area - 1.0)
   g) concavity (severity of concave portions of the contour)
   h) concave points (number of concave portions of the contour)
   i) symmetry

j) fractal dimension ("coastline approximation" - 1)

Our Approaches

Each one of us worked on one of the different datasets above and chose a unique algorithm in order to train our machine learning model. We loaded the datasets and analyzed them and their attributes, pre-processed the data, split it into train and test, and trained the model using our chosen algorithm/approach. We then compared the results and further analyzed them for patterns to try and improve the efficiency of the prediction.

Nikith Rachakonda chose to work on the Diagnostic dataset and used multiple algorithms to train his model, namely, Logistic Regression, Support Vector Machine, and Random Forest.

Aditya Pandey chose to work with the Diagnostic dataset by applying the decision tree classification method. The primary reason to use a decision tree was the fact that a cell is malignant or benign depends on multiple factors that lead to a classification. Decision tree would take all the causative factors to predict whether a cell is benign or malignant.

Tanuj Dave chose to work on the Original and the Diagnostic datasets and took the Gaussian Naive Bayesian approach in order to train the model. The approach seems as the breast cancer tendency will be directly related to the cell and nucleus features and thus, the probabilities of being Benign and Malignant will be dependent upon the probabilities of the features present.

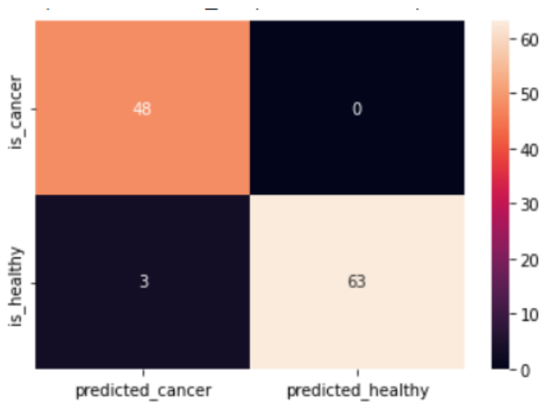
Nikith Rachakonda

Preprocessing

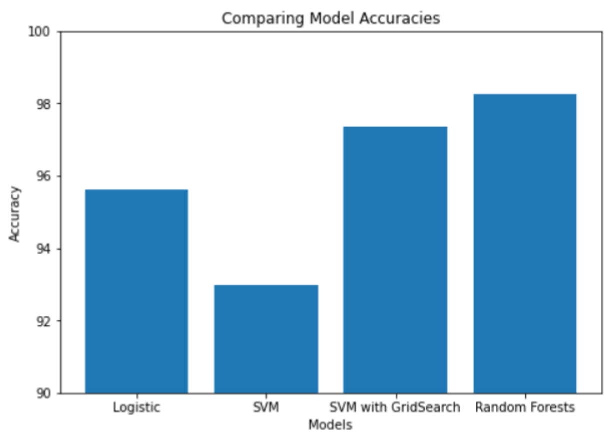
Nikith downloaded the dataset in comma separated value format. Then, he loaded the data from the csv into a Pandas DataFrame. Out of the 32 attributes available, only 30 had any meaningful information – two columns, namely, the ‘id’ and ‘Unnamed:32’ columns, had negligible data. He dropped these two columns, ‘id’ and ‘Unnamed: 32’, because ‘id’ is not required for his classification model and ‘Unnamed: 32’ has no values. For the purpose of his classification model, he also changed the values in the diagnosis column from ‘M’ and ‘B’ to 1s and 0s so that it would be easier to perform testing. For the purpose of demonstrating Support Vector Machines, the data was standardized using MinMaxScaling to transform the attributes to normal distribution. Apart from this, there was no other preprocessing required because the dataset provided is already clean and doesn’t have any missing values.

Models and Evaluation

Nikith first used Logistic Regression for his classification model. It was done using the sklearn.linear_model library. He used the liblinear solver for this model because it is a better choice for small datasets such as this one, giving him an accuracy of 95.61%. Next, he tried using Support Vector Machines on the normalized data. It was done using the sklearn.linear_model library. After fitting the model and creating a confusion matrix for the Support Vector Machine, he got an accuracy of 92.98%. Because he got a much lower accuracy rate than the Logistic Regression, Nikith then tried using GridSearch to find the best parameters for the Support Vector Machine. The best parameters he found were: (C=10) which is the regularization parameter, (kernel = ‘rbf’) which specifies the rbf kernel, and (gamma = 0.1) which is the kernel coefficient for the SVC algorithm in sklearn. After finding the best parameters, he trained the Support Vector Machine again with the new parameters and achieved an accuracy rate of 97.37%. Lastly, he used Random Forests on the unscaled data, and got an accuracy rate of 98.25%.



The above confusion matrix shows the result of his SVM model after using GridSearch to find the best parameters. The label on the left shows the actual values for whether a person had cancer or not, and the label on the bottom shows whether the model predicted the person has cancer or not. From the matrix, it is evident that the model has done a decent job at that, by predicting 48 malignant cases of cancer correctly and also predicting 63 benign cases of cancer. There were 3 cases where the model inaccurately predicted a patient to have a malignant tumor but in reality, they were very much healthy. So, while it was not perfect, the model still gave us a pretty reasonable error margin.

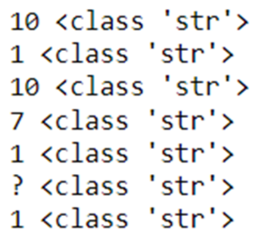


From the above bar plot, it is pretty evident that Random Forests performed best, with an accuracy of around 98%. Logistic Regression, while initially having a better accuracy than the SVM, was bested by the Support Vector Machine after using GridSearch, which was second best. Using GridSearch was a good idea because the updated Support Vector Machine trumped the initial Support Vector Machine by a good margin, so clearly it is important to fine tune the parameters for an algorithm for it to perform at its highest level.

Gaussian Naive Bayes (Tanuj Dave)

Preprocessing

In the initial analysis of the datasets, it was observed that both of them had almost double benign entries than malignant, and also had an id field which was irrelevant to the study and thus removed. The Diagnostic dataset has no missing entries and it was loaded using pandas. Standardization was applied due to the wide range of the real values using the StandardScaler() method. Dataframe 1 or Diagnostic dataset has M for malignant and B for benign which was changed to 0 and 1 since this binary approach yielded faster results.

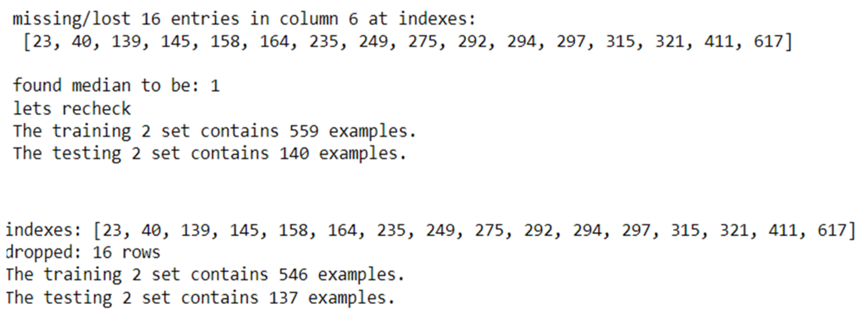


```
10 <class 'str'>
1 <class 'str'>
10 <class 'str'>
7 <class 'str'>
1 <class 'str'>
? <class 'str'>
1 <class 'str'>
```

Fig: Bare Nuclei column with integers as strings and '?'

The Original dataset (dataframe2) contained all integer values from 1 to 10, I applied the same preprocessing as dataset 1 (diagnostic), the attributes for malignant and benign were 2 and 4 which were changed to 0 and 1. The dataframe2 loaded using pandas had some string entries which contained either an integer or a '?', thus, the entries with integer string were converted to integers. The dataframe 2 also had some missing entries, initially there were 2 approaches taken, taking the median of the column and replacing the missing entries with the median and second was removing the entries. The '?' was also replaced with the median of the column. The 2 approaches yielded different results which are discussed later.

The figure below shows that missing entries were found only in column 6 (Bare Nuclei) and their indexes. Then the median approach was taken to find the median to replace the missing entries. Then the removal approach was taken to remove the entries with missing values, which dropped 16 rows total from the data.



```
missing/lost 16 entries in column 6 at indexes:
 [23, 40, 139, 145, 158, 164, 235, 249, 275, 292, 294, 297, 315, 321, 411, 617]

found median to be: 1
lets recheck
The training 2 set contains 559 examples.
The testing 2 set contains 140 examples.


indexes: [23, 40, 139, 145, 158, 164, 235, 249, 275, 292, 294, 297, 315, 321, 411, 617]
dropped: 16 rows
The training 2 set contains 546 examples.
The testing 2 set contains 137 examples.
```

The dataframes were split using the train_test_split() method in the sklearn library using 80-20, 70-30 and 60-40 split for train and test.

Training and Testing

The Gaussian Naive Bayes approach was taken to train the model as the approach seems as the breast cancer tendency will be directly related to the cell and nucleus features and thus, the probabilities of being Benign and Malignant will be dependent upon the probabilities of each of the features present.

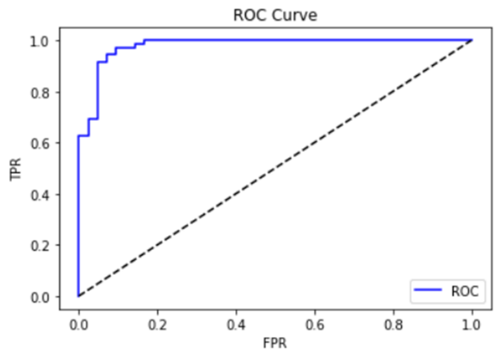
GaussianNB() method of the sklearn library was used to obtain a model and the fit() method was applied to fit the training data. The ROC curves were plotted to better understand the prediction and accuracy of the model, the class probabilities were obtained from the model using the predict_proba() method. The predict() method was used to predict the labels of testing data. plot_roc_curve() method developed during the class was then used to plot the roc curve for each combination of the train-test split and preprocessing method.

Evaluation

For the Diagnostic dataset, gaussian naive bayes worked well and had an accuracy of 94.7 percent on an 80-20 train-test . It mispredicted only 6 out of 114 testing samples. The ROC had 0.979 area under the curve, area close to 1 shows that the curve is nearly ideal and the model has a good True positive to False positive ratio. There was a good improvement from 92 percent to 94 percent when the split was moved from 60-40 or 70-30 to 80-20 train test split, no significant improvement was seen by increasing train size from 80%. The image below shows the ROC curve for Diagnostic dataset.

```
Number of mislabeled points out of a total 114 points : 6

accuracy of GNB is 94.73684210526315 %
```



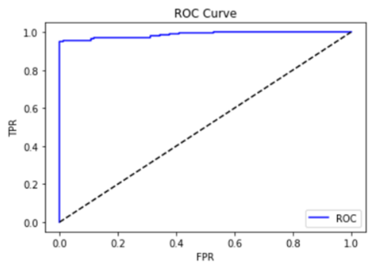
```
0.9788359788359788
```

Fig: Prediction and ROC curve for Diagnostic Dataset

The Original dataset using the median replacement approach gave a 96.4 percent accuracy on an 80-20 train-test split. Only 5 out of 140 testing samples were mispredicted. The ROC curve had 0.985 area under the curve, which as is closer to 1 and combined with the accuracy shows that the model fit very well and had a great True to False positive ratio. The images are included below.

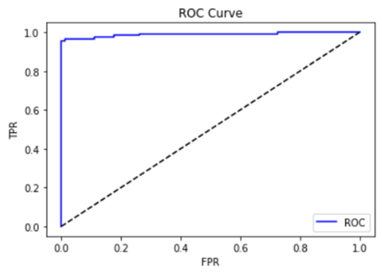
The removal approach on the Original dataset on an 80-20 train-test split gave 100 percent accuracy. 0 of 137 samples were mispredicted. The ROC curve had 1 area under the curve which combined with the 100 percent accuracy shows that the model is ideal. This behavior though ideal raised ambiguity about the removal approach and thus different train-test split methods were tried and the results are shown below.

```
Number of mislabeled points out of a total 280 points : 1
accuracy of GNB is 96.42857142857143 %
```



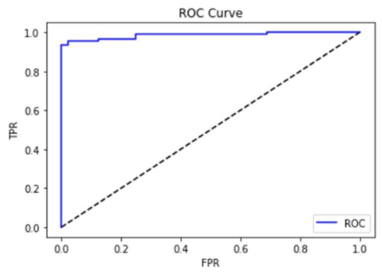
```
0.987637283652464
```

```
Number of mislabeled points out of a total 210 points : 6
accuracy of GNB is 97.14285714285714 %
```



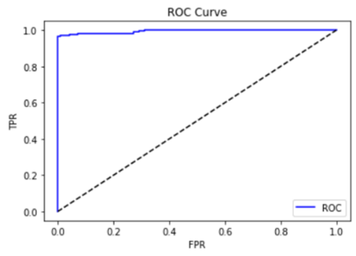
```
0.9898010198980102
```

```
Number of mislabeled points out of a total 140 points : 5
accuracy of GNB is 96.42857142857143 %
```



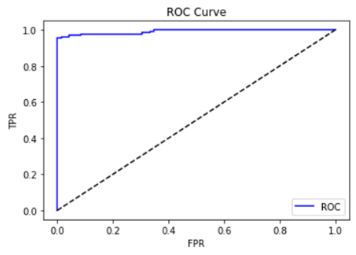
```
0.9852807971014492
```

```
Number of mislabeled points out of a total 274 points : 7
accuracy of GNB is 97.44525547445255 %
```



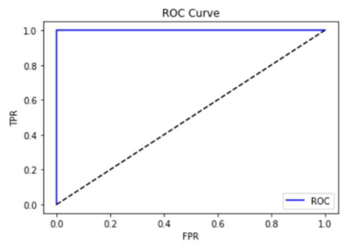
```
0.9943820224719101
```

```
Number of mislabeled points out of a total 205 points : 6
accuracy of GNB is 97.07317073170731 %
```



```
0.9915413533834587
```

```
Number of mislabeled points out of a total 137 points : 0
accuracy of GNB is 100.0 %
```



```
1.0
```

Fig: 60-40, 70 - 30, 80 - 20 split median approach above, removal approach below.

Aditya Pandey (Decision Tree Classification)

Research Method

Aditya has used the pandas, NumPy, and sklearn library. Aditya imported the dataset by using pandas.read_csv() method to read from a URL and assign features their respective name. Aditya split the data into a 60% training set and a 40% testing set. Used pandas.read(). Certain features had a question mark in their sample(“?”) instead of a value. To replace the question mark, He utilized np.nan operation to convert that sample’s value to Not-A-Number. Performed the pandas.nan() operation on all sample values by looking them up with the help of pandas.read_csv.replace(to_replace=”?”, value = np.nan) where the first parameter is the value to be searched in the feature table, and the second is the value to be replaced with. After replacement of the question mark, He performed pandas.dropna() which removes the entire row if a column contains Not-A-Number.

Decision Tree Classification

I used the sklearn's decision tree classifier library to classify the data. The library works by calculating the Entropy of each feature by

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$where\ p_i\ is\ the\ count(True)\ or\ count(False)\ for\ each\ feature\ S$$

Then the entropy is plugged into the formula:

$$Gain(S,A) = E(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|}\ where\ v\ is\ the\ value\ of\ the\ sample$$

Finally,

$$Root = arg\max\left(Gain(S,A)\right)$$

The classifier loops through all the gains and finds the maximum gain which becomes the root of the tree. Following which, the function uses the calculated feature's splitting criteria to build the tree; it compares the sample value to the splitting criteria. The information gain determines the quality of the split.

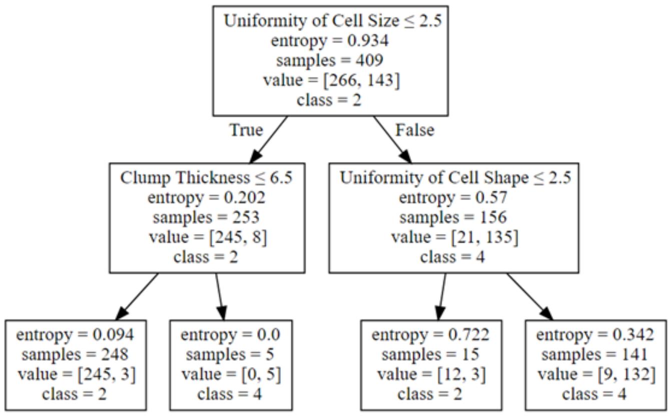
Sklearn's library handles all the calculations and evaluation, but the logic is the same. For my dataset, sklearn's decision tree classifier chose *Uniformity of Cell Size* as the splitting criteria for the tree.

After building the root node and its child node, the tree again calculates the information gain from all the sample and choses the next splitting criteria. After building each successive splitting criteria, the tree eventually reaches the leaf node. The leaf node contains the binary class which it returns. The binary class value is the prediction value which the user gets after it walks down the tree by choosing the path based on the splitting criteria.

After all the calculations, I created a function the plots the decision tree. I plotted the tree for a maximum depth of 2 to account for screen size constraints. The plot will be explored in the modeling section.

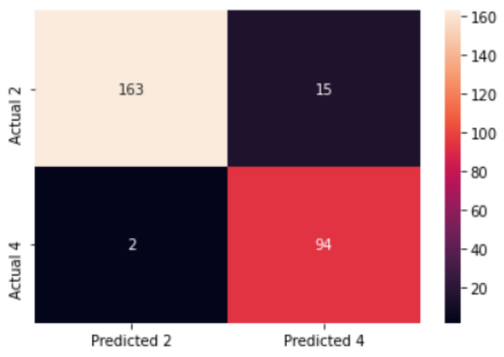
Modelling and Analysis

The Image on the right illustrates the split that the decision tree classifier generated. As we can see over here, the classifier used different splitting criteria to calculate the quality of the split. Hence, difference features are mapped to form a path that leads to the binary class value. A class value of 2 represents that the cell is benign whereas a class value of 4 represents that the cell is malignant. Furthermore, as soon



as we reach the leaf nodes, the entropy becomes zero due to the fact that there are no more samples to classify. Nevertheless, tree has not finished building after depth of 2 because we still have some more entropies to process. Until all the leaf nodes contain an entropy value of zero, the tree will continue to split.

The confusion matrix obtained on the right shows the accuracy of the prediction value. It was able to predict the benign (Class value 2) correctly 163 times and got it wrong only 15 times. The prediction for malignant (Class value 4) was also comparatively accurate because it was able to predict malignancy 94 times and it only got a wrong prediction 2 times. Overall, the accuracy after all the computations came to a 94.16% which is reasonable considering that I used a 60% train and 40% split. Hence, the classifier did not have enough data to train itself.



Conclusion

In conclusion, the results met Nikith's assumptions that Random Forests would perform the best and that SVM would provide reasonable results. While initially SVM had done poorly compared to the others, after fine-tuning some parameters, it met his expectations. He has since learned that choosing the right dataset with enough data is vital for a machine learning project. A larger dataset might have given Nikith the opportunity to work with more algorithms and parameters that would provide the best result. Further improvements could probably be made by using techniques such as Ensemble Learning.

The removal approach worked best for the Gaussian Naive Bayes with an 80-20 split with a surprising 100% accuracy. While the 70-30 split also has similar accuracy for median approach, remove worked the best of two for the Original dataset. 80-20 split using Gaussian Naive Bayes has a 94.7% accuracy.

Thus, the Random Forest Approach gave the best results at a 98.25% accuracy compared to 94.7% for Naive Bayes and 94.16% for Decision Trees. Out of the other two Naive Bayes gives better results for the Breast Cancer prediction as it takes into account the conditional probabilities rather than splitting criteria for the cell features.

Some further improvements to the model could be using a larger dataset with less noise to optimize fitting the prediction. Furthemore, training the dataset using a single layer perceptron with a sigmoid activation function will improve the accuracy even more as it will be able to classify the test dataset better.

Generally, the accuracy of the different approaches–to model breast cancer prediction–that we presented in this paper is high due to the fact that the dataset is used extensively in industry and academica by scholars and professionals which would result in a 'higher quality data'. Real world testing with a different dataset that has not been processed in such a formal setting may lead to a different accuracy score, and a different confusion matrix.

<u>References</u>

UCI Machine Learning Repository: Breast Cancer wisconsin (original) data set. (n.d.). Retrieved May 5, 2022, from https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28original%29

Wolberg, W. H., Street, N., & Mangasarian, O. L. (1995, November 1). *Breast Cancer Wisconsin (Diagnostic) Data Set*. UCI Machine Learning Repository: Breast Cancer wisconsin (diagnostic) data set. Retrieved May 5, 2022, from https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.

1. O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18.

2. William H. Wolberg and O.L. Mangasarian: "Multisurface method of pattern separation for medical diagnosis applied to breast cytology", Proceedings of the National Academy of Sciences, U.S.A., Volume 87, December 1990, pp 9193-9196.

3. O. L. Mangasarian, R. Setiono, and W.H. Wolberg: "Pattern recognition via linear programming: Theory and application to medical diagnosis", in: "Large-scale numerical optimization", Thomas F. Coleman and Yuying Li, editors, SIAM Publications, Philadelphia 1990, pp 22-30.

4. K. P. Bennett & O. L. Mangasarian: "Robust linear programming discrimination of two linearly inseparable sets", Optimization Methods and Software 1, 1992, 23-34 (Gordon & Breach Science Publishers).