MACHINE LEARNING ENGINEER NANO DEGREE

BREAST CANCER PREDICTION MODEL

By: Mahmoud Galal Ahmed Abo Hamda
23-1-2019

|. DEFINITION

Project Overview

Breast cancer is one of the most commonly spreading types of cancer between women in my home country Egypt and worldwide, and it affected a close people to me, and as this article and the Breast Cancer Foundation of Egypt (BCFE) say that the breast cancer incidences in Egypt has grown to 32% percent and it's predicted to grow more as the population grows. So it's a very important problem to solve and I personally would like to make it my project subject. And it will be a classification model that predicts if the patient have a benign or malign tumor that needs to be cured or examined carefully again, and we can find the dataset for this project here.

Problem statement

The problem as we said in the previous section is the growing number of the breast cancer incidences in Egypt and worldwide and the difficulties in diagnosing the tumor early. The solution will be a model to detect if the tumor is malign or benign to ease up the process of identifying the disease and doctors can diagnose the features of the tumor and then they can use a mobile, web or desktop application to send these features to an API that will ask the model and return results if the tumor is malignant or not, then the doctor can take the right procedures to cure the tumor as soon as possible to avoid further spreading of the tumor in other parts of the body.

Metrics

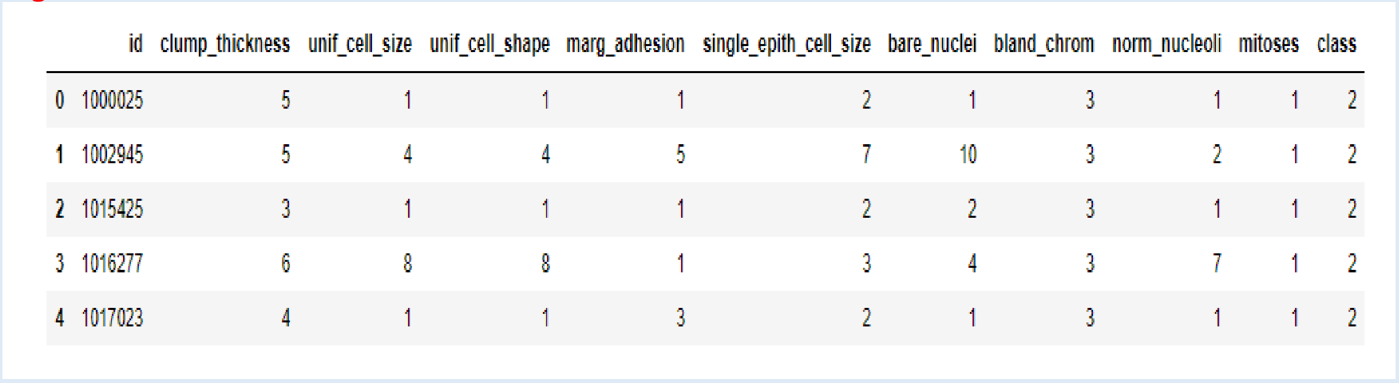
I'll use f-beta score and will make it close to the recall part of the line as we should be interested more in the false negative area of the confusion matrix.

II. ANALYSIS

Data exploration

- The dataset has features of the tumor and the class of it as a label and here is a sample from the data set:

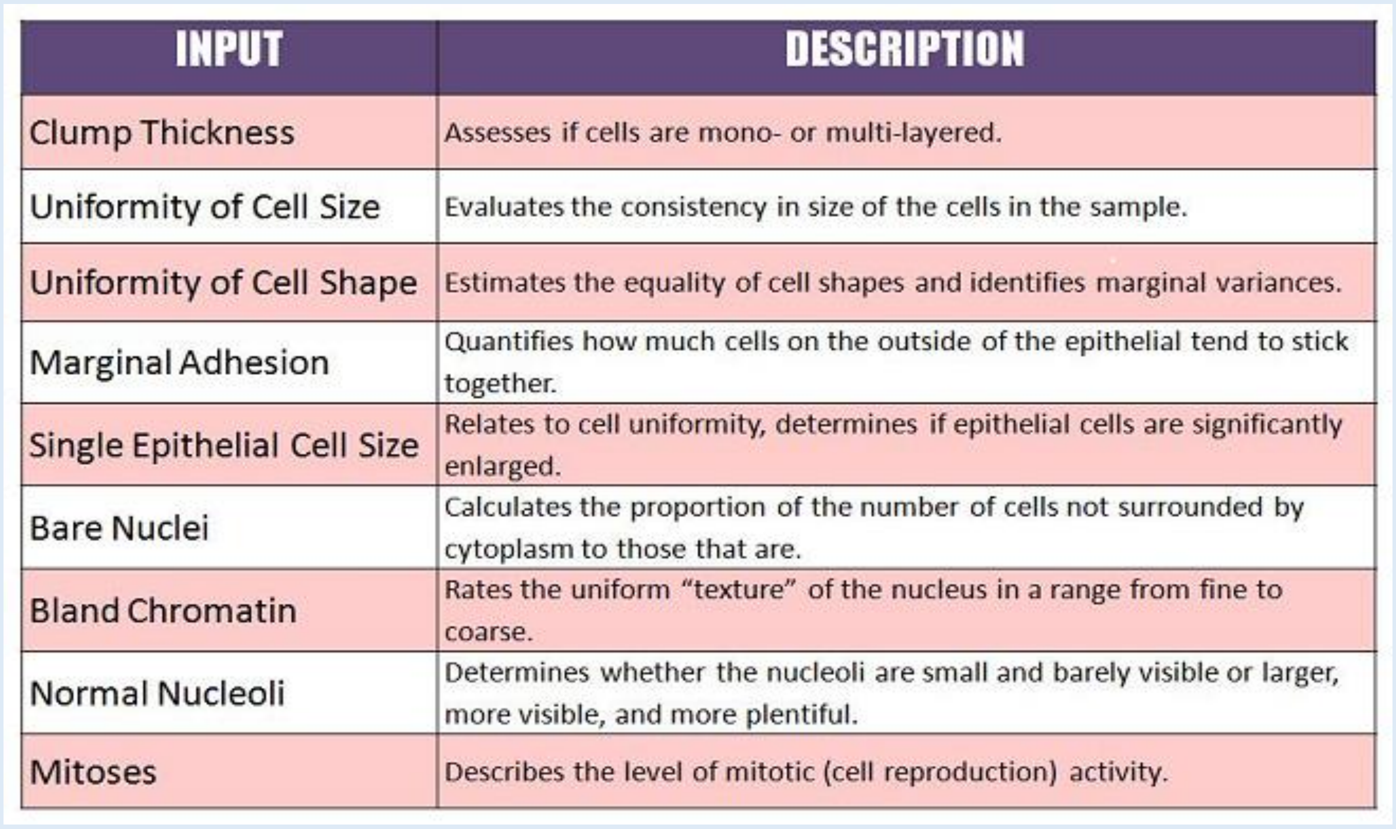
Figure 1:



	id	clump_thickness	unif_cell_size	unif_cell_shape	marg_adhesion	single_epith_cell_size	bare_nuclei	bland_chrom	norm_nucleoli	mitoses	class
0	1000025	5	1	1	1	2	1	3	1	1	2
1	1002945	5	4	4	5	7	10	3	2	1	2
2	1015425	3	1	1	1	2	2	3	1	1	2
3	1016277	6	8	8	1	3	4	3	7	1	2
4	1017023	4	1	1	3	2	1	3	1	1	2

And I found the description of each of these features here and approved by a doctor friend:

Figure 2:



INPUT	DESCRIPTION
Clump Thickness	Assesses if cells are mono- or multi-layered.
Uniformity of Cell Size	Evaluates the consistency in size of the cells in the sample.
Uniformity of Cell Shape	Estimates the equality of cell shapes and identifies marginal variances.
Marginal Adhesion	Quantifies how much cells on the outside of the epithelial tend to stick together.
Single Epithelial Cell Size	Relates to cell uniformity, determines if epithelial cells are significantly enlarged.
Bare Nuclei	Calculates the proportion of the number of cells not surrounded by cytoplasm to those that are.
Bland Chromatin	Rates the uniform "texture" of the nucleus in a range from fine to coarse.
Normal Nucleoli	Determines whether the nucleoli are small and barely visible or larger, more visible, and more plentiful.
Mitoses	Describes the level of mitotic (cell reproduction) activity.

And each of these features can contribute in the classification in the following way:

1- **clump_thickness**: (1-10). Benign cells tend to be grouped in monolayers, while cancerous cells are often grouped in multilayers.

2- **cell_size_uniformity**: (1-10). Cancer cells tend to vary in size and shape. That is why these parameters are valuable in determining whether the cells are cancerous or not.

3- **cell_shape_uniformity**: (1-10). Uniformity of cell size/shape: Cancer cells tend to vary in size and shape. That is why these parameters are valuable in determining whether the cells are cancerous or not.

4- **marginal_adhesion**: (1-10). Normal cells tend to stick together. Cancer cells tends to lose this ability. So loss of adhesion is a sign of malignancy.

5- **single_epithelial_cell_size**: (1-10). It is related to the uniformity mentioned above. Epithelial cells that are significantly enlarged may be a malignant cell.

6- **bare_nuclei**: (1-10). This is a term used for nuclei that is not surrounded by cytoplasm (the rest of the cell). Those are typically seen in benign tumors.

7- **bland_chromatin**: (1-10). Describes a uniform "texture" of the nucleus seen in benign cells. In cancer cells the chromatin tend to be coarser.

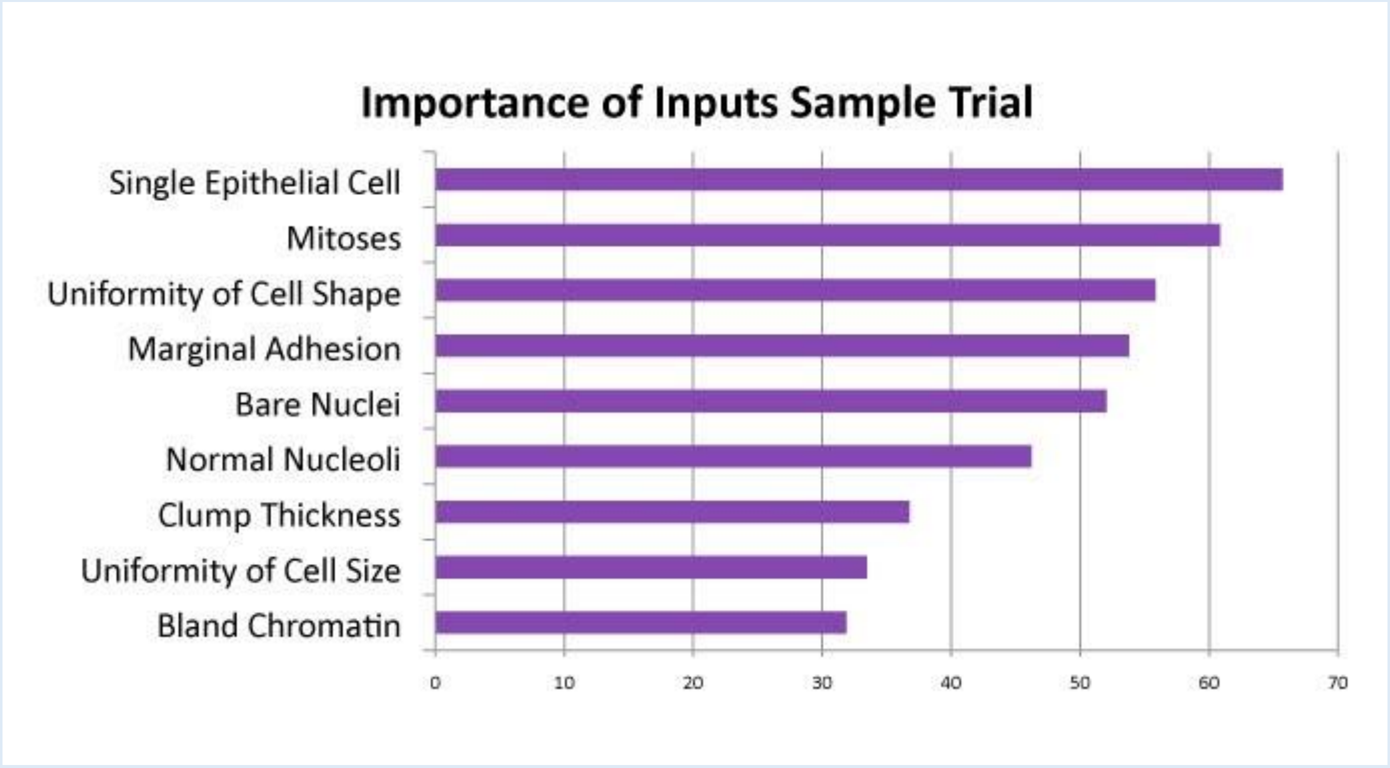
8- **normal_nucleoli**: (1-10). Nucleoli are small structures seen in the nucleus. In normal cells the nucleolus is usually very small if visible at all. In cancer cells the nucleoli become more prominent, and sometimes there are more of them.

9- **Mitoses**: (1-10). Cancer is essentially a disease of uncontrolled mitosis.

All the features description above is taken from Neural designer website.

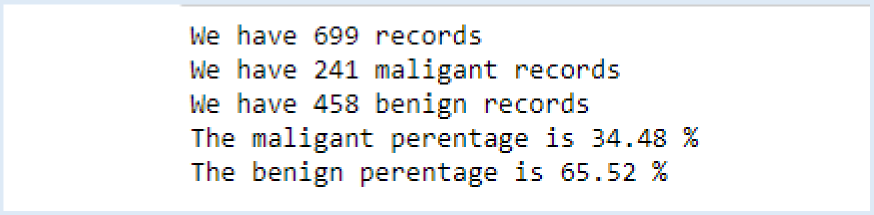
And we can order the features by its importance according to this graph:

Figure 3:



This figure and further analysis is taken from here.

And we can see the statistics about this data set as follows:



```
We have 699 records
We have 241 maligant records
We have 458 benign records
The maligant perentage is 34.48 %
The benign perentage is 65.52 %
```

So the percentage of the benign cases is approximately double the size of the malignant one

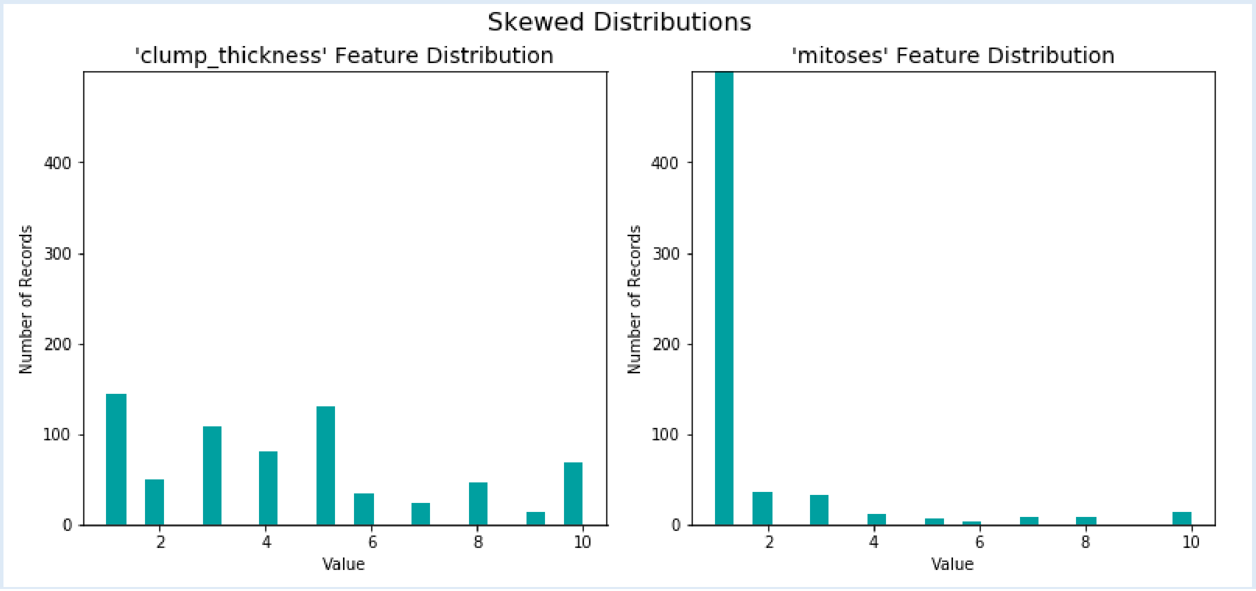
And the input size would be 559 records as a training samples and 140 records as a testing samples.

There are some records on the ‘bare_nuclei’ column that have the ‘?’ symbol instead of a number, so I replaced it with a very large negative value that then would be scaled at the scaling processing of the data in order to avoid the effect of outliers in our dataset.

Exploratory visualization

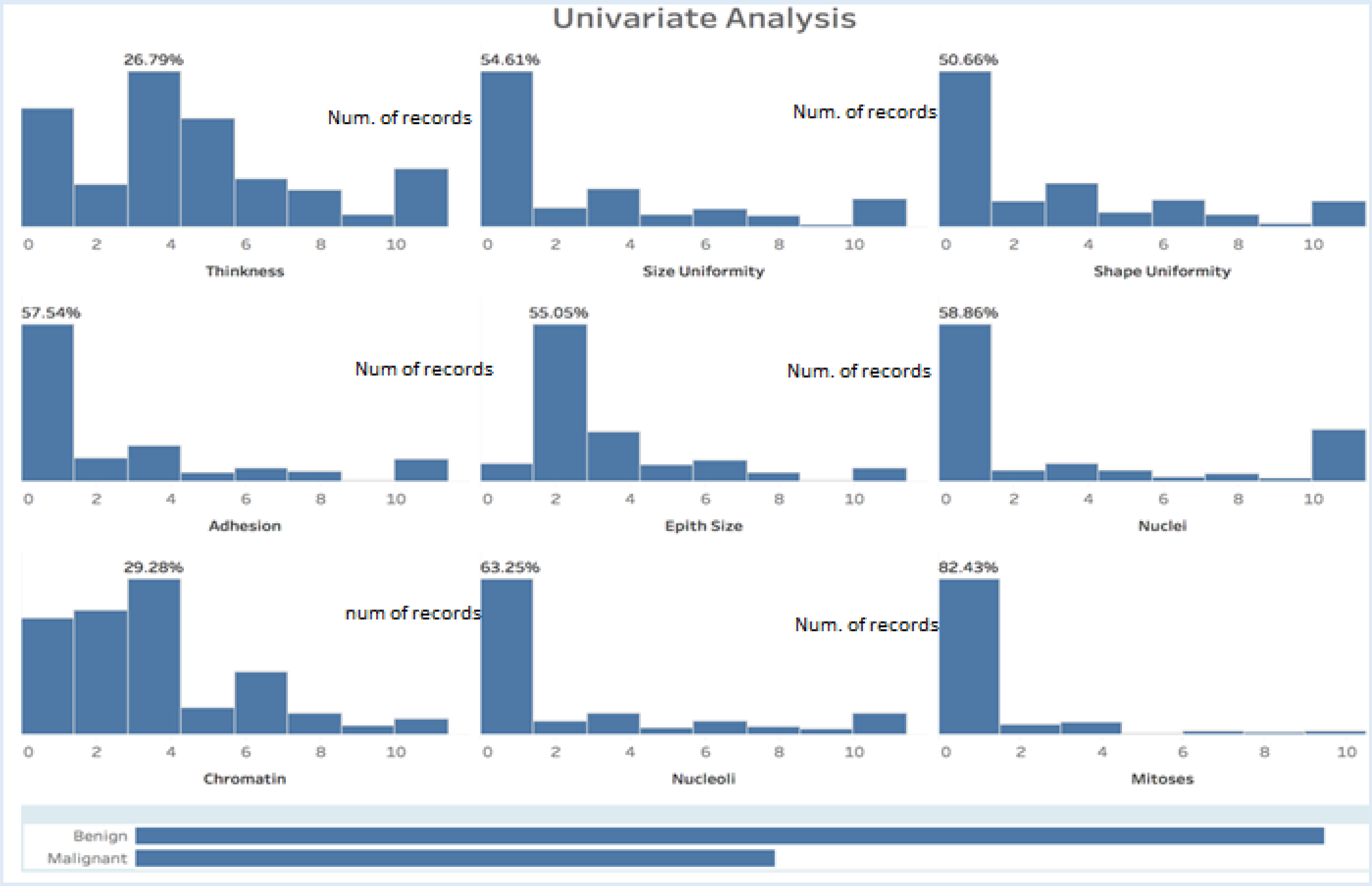
I have tried to find a way of finding whether I have skewed features or not so that I choose the scaling process based on that and I did the visualization on my features like:

Figure 4:



So we see that the 'mitoses' is skewed around the 1, so I have chosen the min-max scaller due to its effectiveness in dealing with skewed data and outliers as it scale the data between two intervals and I choosed the default one which is between 0 and 1, and it's important to visualize our data to see if the cleaning process is required, so the visualization part is very important to not be working in the dark especially in a sensitive topic like breast cancer.

Figure 5:



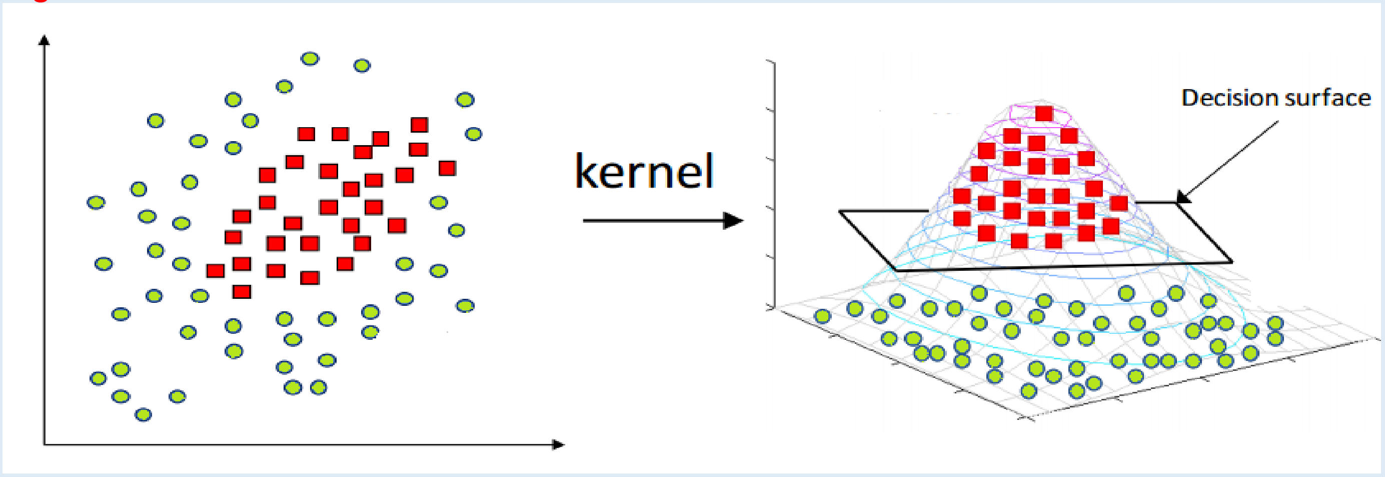
And this figure shows all the features and the y-axis represents the number of records like figure 3.

Algorithms and techniques

I have tried several algorithms such as SVM with a linear kernel because the data is linear and SVM is an algorithm that analyzes large amount of data to identify patterns from them and it creates hyper planes that have the largest margin in a high dimensional space to separate given

data into classes. The margin between the 2 classes represents the longest distance between closest data points of those classes
SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form.

Figure 6:



and also tried the logistic regression due to its robosty to data with two classes and linear data and also its ability to work with dependent features which we want to
Infer from it a yes or no answer, and also it's so fast in training, test and prediction processes and it works as follows:

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

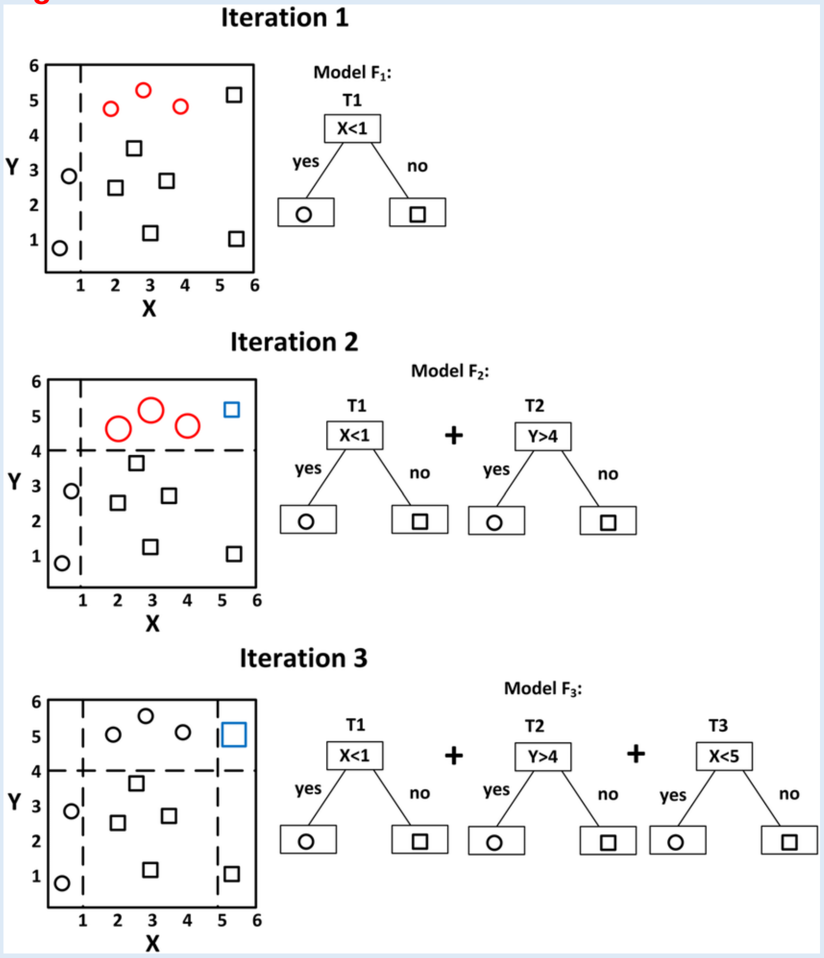
Below is an example logistic regression equation:

$$y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})$$

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

But the previous one has been used before in the benchmark model (Look at the benchmark section) that I have included in my proposal, so I tried a different great algorithm which is Gradient boosting, and here is a figure of how it works:

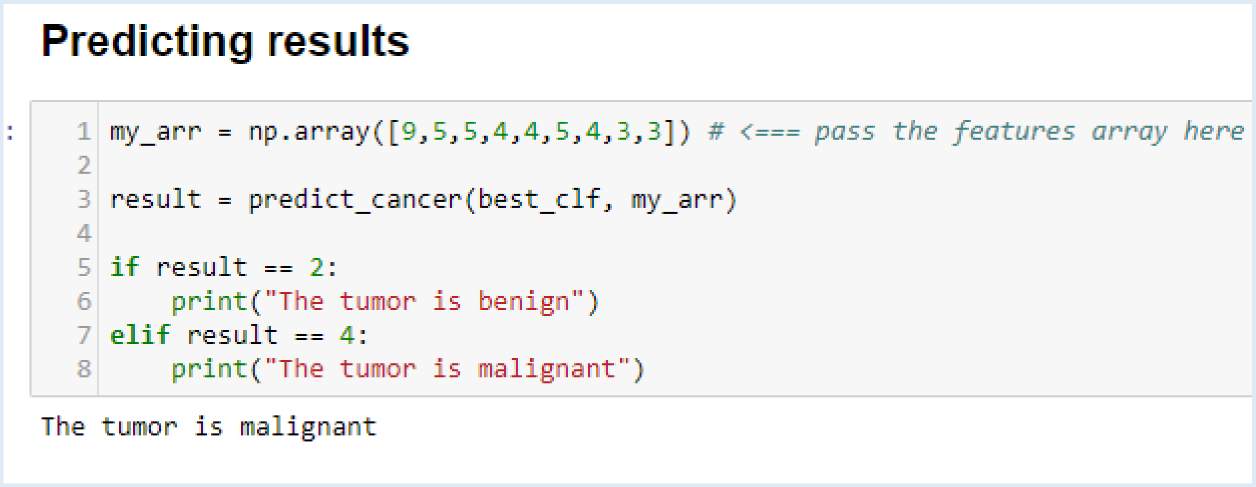
Figure 7:



Gradient boosting works sequentially, so it learns from the past errors and optimize it as they are called pseudo-residuals, so it returns great precise results and it applies the concept of boosting and its weak learners consists of decision trees and it’s a good candidate for our data as it's clean, well distributed and sufficiently large and as I have done the data scaling and cleaned up my data from noise records, I thought it will be great for that case.
And it applies the concept of boosting and its weak learners consists of decision trees, conversely random forests applies the bagging concept, and as gradient boosting works in subsequent predictors learn from previous ones, so it takes less time and also become more accurate than random forests as it works in parallel way then combines the results using many techniques such as weighted average, majority vote and normal average.

The hyper parameters of the gradient boosting algorithm have been chosen by the grid search technique to return the best possible estimator for our data.

And the tester can pass the input to the prediction function and it will return the result:



Predicting results

```
my_arr = np.array([9,5,5,4,4,5,4,3,3]) # <=== pass the features array here

result = predict_cancer(best_clf, my_arr)

if result == 2:
    print("The tumor is benign")
elif result == 4:
    print("The tumor is malignant")
```

```
The tumor is malignant
```

Benchmark

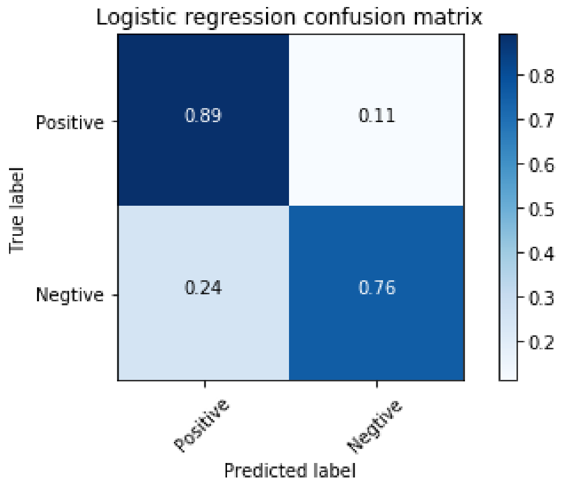
My benchmark is a logistic regression model that meant to work in the same way to solve our problem as my model (Gradient boosting), but the difference is that I'll try to beat its score with gradient boosting.

The benchmark model gave a result of 87.7% and plotted its confusion matrix scores as shown below:

Figure 8:

```
Logistic regression fbeta score 87.72

Normalized confusion matrix
[[0.89010989 0.10989011]
 [0.24489796 0.75510204]]
```



I have put the benchmark model file that contains all the source code in the project files.

|||. METHODOLOGY

Data processing

The gradient boosting algorithm needs the data to be clean and out of noise and it would be great if it was scaled as it takes longer in the training time than other algorithms so if the data was scaled it will decrease the time taken on the training process and that's what we did to our data:

- First we have excluded the records that have the '?' symbol instead of its number, as that noise will surely affect out results.

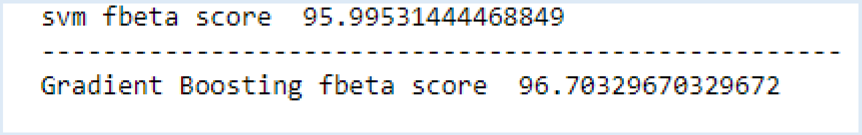
- Second we scaled our numbers between 0 and 1with the min-max scaller.

- Then we splitted the data randomly (as the shuffle parameter is True by default) into training and testing sets, and assigned the testing set to be 20% of the whole records which is 140 samples and the other 559 samples were training set, and also assigned a number to the random state parameter of the train_test_split function to reserving the state of the that specific splitting process.

Implementation

- First I have imported the essential libraries required for the project like numpy, pandas and display from Ipython.
- Then I have read the data from the file ('breast-cancer-wisconsin.data.txt') using pandas' read_csv function, and then displayed the first 5 records.
- Then I wanted to get some statistics about the data, so I have called the pandas data frame's describe method to display me some stats.
- Then I did some data preparation like replacing invalid entries that have '?' instead of a number with a dummy value that will be scaled after then.
- Then I removed the id column as it will affect the results and it has no importance here.

- Then I splitted the features in the variable X from the label variable Y and printed some information about the data like the number of malignant and benign records and the percentage of each one of them.
- Then I have visualized some features to see their distribution form.
- Then I used train_test_split function to split the data into training and testing sets and made the testing set to be 20% of the whole set.
- Then I imported MinMaxScaler to do some data scaling in order to optimize my model performance, and applied the scaling on my training and testing features set.
- Then in the model selection part I set to candidates to beat the benchmark model and they are the SVM algorithm with parameters (kernel = 'linear', random_state = 5) and Gradient boosting with parameters (random_state=70), so I initialize the classifiers after importing their classes from sklearn library then we trained our models with the training sets and then got the predicted results from each one of them.
- Then we went to the model evaluation process to see which one of them will be the challenger for the benchmark model, so we imported the evaluation metric which was fbeta and evaluated each model by comparing the results of each one with the actual labels and printed the results, and I found out the gradient boosting score is higher as shown:



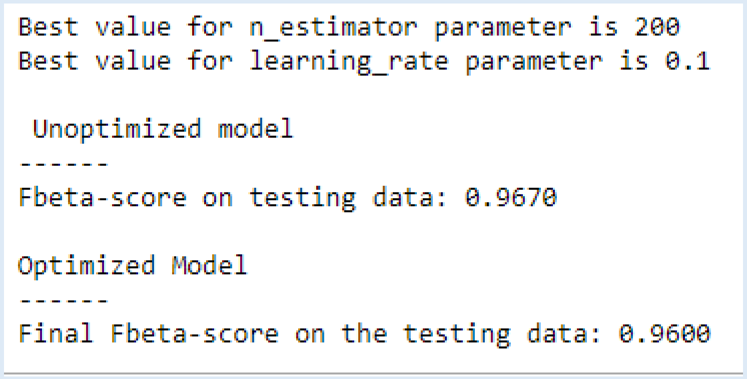
```
svm fbeta score  95.99531444468849
-----------------------------------------------------
Gradient Boosting fbeta score  96.70329670329672
```

- Then i choosed gradient boosting to represent my model, and began to refine this model using Grid search technique, so I imported gridsearchCV, shuffle_split for the cross validation, make scorer to create the scoring function and fbeta_score metric to provide it to the make scorer function, then I initialized all the parameters needed for the grid search class like the classifier, the set of parameters (n_estimators with values [100, 200, 400] and learning rate with values starts small like [0.01, 0.1, 1.5]) , cv_sets and made 10 splits that each split have 20% percent of it as testing sets and finally the scorer parameter that I have passed to it the fbeta_score function, then I obtained the best classifier after training the grid search object and used the best estimator to compare between gradient boosting performance before and after the optimizing process and the score decreased with small fractions, but it was still beating the SVM and benchmark model and I'm sure that as long as the data grows the model performance will do better and generalize more and hence will become more and more robust, and I encountered some difficulties in choosing the parameters values to pass to the grid search and took me a time searching for the best values for each parameter like the learning rate for example, but I managed to find very good results without going to overfitting problem as I used preventing techniques like grid search and cross validation.
- Then I have declared a function for the tester to use, and it will accept a classifier handle and array of features and will reshape the array to make it valid for the classifier object (it will be in this form [[1,2,3,3,3]]), and then returning the result.

- The final code cell is the testing process that invokes the function and interpret its result, if the result is 2 then the tumor is benign, and if it was 4 so the tumor will be classified as malignant.

Refinement

- First I have removed the noise from the data as discussed above
- Then I have scaled the data to obtain a better performance
- Lastly I used the grid search technique to the gradient boosting classifier to get the best hyper parameters, and I recorded the results before and after getting the best estimator, and i initialized all the parameters needed for the grid search, the set of parameters was n_estimators with values [100, 200, 400] and learning rate with values starts small like [0.01, 0.1, 1.5], and after getting the best estimator from the grid search technique I printed the selected results and it was like:



```
Best value for n_estimator parameter is 200
Best value for learning_rate parameter is 0.1

 Unoptimized model
------
Fbeta-score on testing data: 0.9670

Optimized Model
------
Final Fbeta-score on the testing data: 0.9600
```

|V. RESULTS

Model evaluation and validation

As we discussed in the refinement section that we obtained the final model by getting the best estimator by performing grid search technique with cross validation and after this improvement we got a score higher than 90% as shown in the previous figure and best model choosed 200 for the number of estimators and 0.1 for the learning rate parameter and these values are consistent with the data set as it’s not too big for large number of estimators to learn from it and also not too small, so I think that 200 is a correct value and the learning rate wasn’t too small so the model does not memorize the data while it’s learning and also not big so the model not skip the characteristics of the data and that will cause a huge problem as the model serve a very important aspect that relates to human life.

And by this we can say that out model is appropriate for solving this problem as the gradient boosting is also very robust against larger data set, in fact the larger the data set the better it performs.

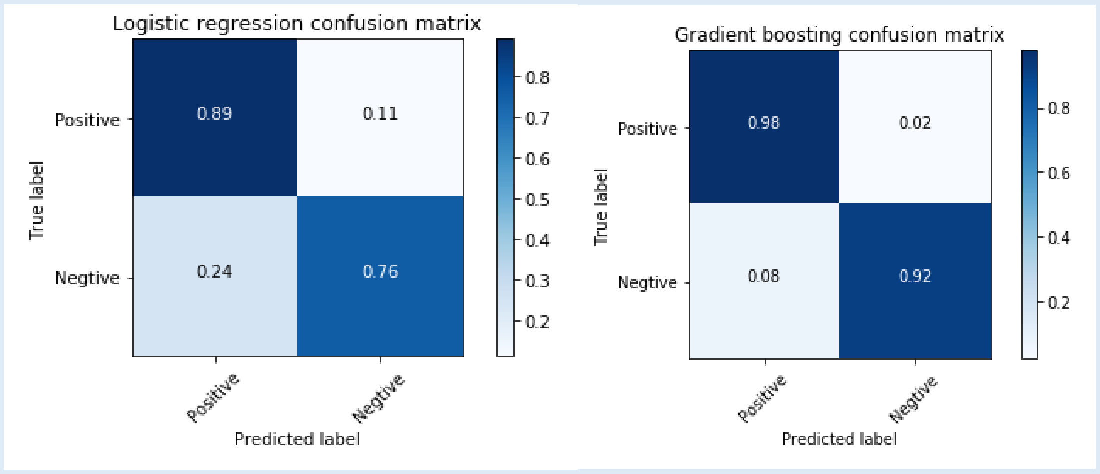
Also we can say that the results of the model have been confirmed by a doctor and we found that the model is very accurate and we tested it with many input samples and it did great.

Justification

The final model (Optimized gradient boosting) is better than the benchmark model, so our model got 96% score while the benchmark was only 87.72% and I think 96% is the highest we can trust here and also I consider the ensemble methods is the strongest models we can work with, so our model is significant enough to deal with this problem.

V. CONCLUSION

Free-form visualization



In the figure 9 above we can see that the final (tuned) gradient boosting is so much better than the benchmark model and after we proved that in all the previous ways like the fbeta score we also visualized the confusion matrix for each one of them and found that we our model is better in the following:

- The true positive is so much higher than the benchmark model by 9%.
- The true negative is so much higher by approximately 16%.
- The precision score (TP / TP + FP) is higher by approximately 14%.
- The recall score (TP / TP + FN) is higher by approximately 9%.

We can see that it's better in every way possible.

Reflection

We can summarize this project in the following simple sentences:

- We got the appropriate data and libraries.
- We cleaned our data to increase the model performance.
- We visualized and reported statistics about the data to get a good grasp of it.
- We choosed the best model and then tuned its hyper parameters to get the best form possible, and then applied this model to real values and tested the results and approved it by a doctor to see that our model's results consistent with its scores that we obtained.

Improvements

- One of the improvements that I'm working on it after my exams as I said before is the data visualization techniques, as in this nano-degree we haven't learn anything about visualization so I have obtained some courses in that field that will help me visualize my data and my results in so many ways that will help in the future projects.
- Also I can provide more data to the input dataset by setting down with more doctors that can give me valid, real data, as with gradient boosting the more data the better it performs.
- I can try different approaches and there are many great new candidates that doesn't have a lot of resources except their documentation and one or two blog posts, so they will need

some study and tries to prove that will be better than out model, and of these models is **LightGBM**, it's a gradient boosting framework that uses tree based learning algorithm, and it's different from the ordinary Gradient boosting in that **Light GBM grows tree vertically** while other algorithm grows trees horizontally meaning that Light GBM grows tree **leaf-wise** while other algorithm grows **level-wise**. It will choose the leaf with max delta loss to grow. When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm, and the difference between leaf-wise and level-wise is like so:

Figure 10:

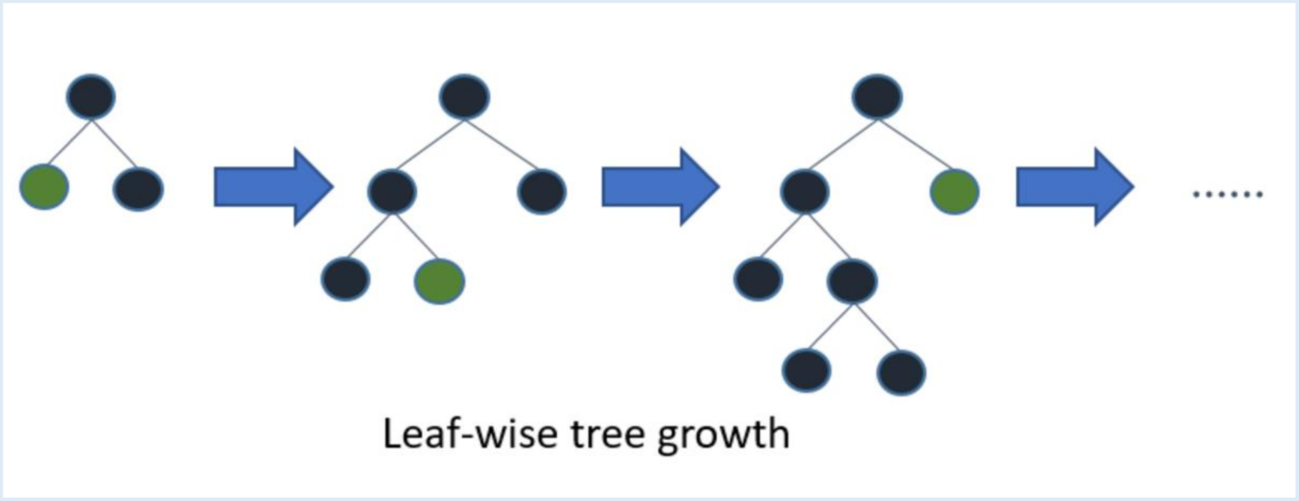
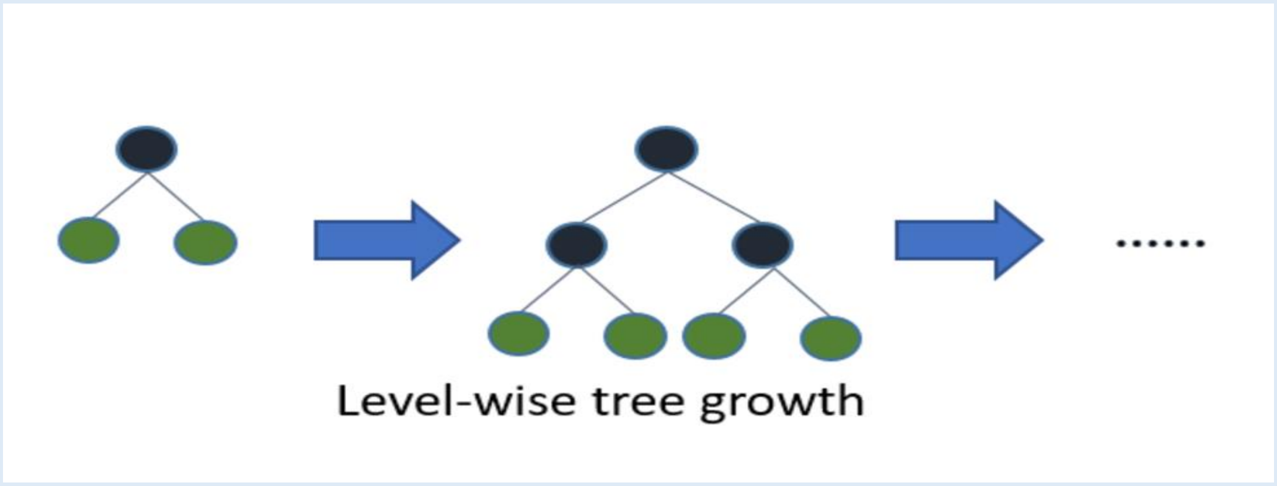


Figure 11:



So we can say that it's extremely powerful as Light GBM is prefixed as 'Light' because of its **high speed.** Light GBM can **handle the large size** of data and **takes lower memory to run**. Another reason of why Light GBM is popular is because it **focuses on accuracy of results**. LGBM also **supports GPU learning** and thus data scientists are widely using LGBM for data science application development, but I think that there is a down side for using this algorithm in our case as the data still needs to get a bit bigger if we wanted to use

LGBM as it is not advisable to use LGBM on small datasets. Light GBM is **sensitive to overfitting** and can easily overfit small data. And upon what I have read There is no threshold on the number of rows but it's suggested to use it only for data with 10,000+ rows, so I can make it as future improvement when I enlarge my data set with verified records.

Note: these figures and most of the information were found at this blog post.

- I can use the Meta ensembling technique to make use of many great and powerful classifiers like SVM, Ada boost, gradient boosting or its new powerful frameworks like LightGBM or XGBoost, and model ensembling technique used to combine information from multiple predictive models to generate a new model. Often times the stacked model (also called 2nd-level model) will outperform each of the individual models due its smoothing nature and ability to highlight each base model where it performs best and discredit each base model where it performs poorly. For this reason, stacking is most effective when the base models are significantly different, so I think this improvement is an important one to study and apply.
- After my exams I can apply the encapsulation technique and hide all the implementation details and provide just a user interface to deal with the model by supplying the parameters in GUI and return the result of the diagnosis to the user, and also to support the transfer learning concept I'll provide the code in a github repository to the public to benefit from and optimize it more and more.
- And of course I believe that there are better solution out there which I can found or anyone else can, but for now I'm really happy and proud of this one.