Aman Ahluwalia

Capstone Project

Machine Learning Nanodegree

11 July 2017

Breast Cancer Diagnosis

Definition

Project Overview

Breast cancer is a complex and heterogeneous disease due to its diverse morphological features, as well as different clinical outcome. As a result, breast cancer patients may response to different therapeutic options. Currently, difficulties in recognising the breast cancer types lead to inefficient treatments. Either it is diagnosed as malignant or benign. Therefore it is necessary to devise a clinically meaningful classification of the disease that can accurately classify breast cancer tissues into relevant classes. This study aims to classify breast cancer lesions which have been obtained from fine needle aspiration (FNA) procedure using the best machine learning practices.

Problem Statement

The goal is to maximise the accuracy in diagnoses based on cytologic features derived directly from a digital scan of fine-needle aspirate (FNA) slides. It clearly sounds like a classic classification problem in machine learning. We will go through different available and relevant machine learning classification algorithms, to solve the problem recognising the breast cancer type. Then we will fine tune the algorithms to maximise the accuracy with which we can judge if the patient is diagnosed as malignant or benign. At last we will try to find the best solution for the above stated problem, so that the patients can be reliably diagnosed before starting any medication.

Metrics

The metrics we will be using to you will use to measure performance of our model is the F_1 score. In statistical analysis of binary classification, the F_1 score (also F-score or F-measure) is a measure of a test's accuracy. The F_1 score can be interpreted as weighted average of the precision and recall, where an F_1 score reaches its best value at 1 and worst at 0.

Precision is defined as, out of all the items labelled as positive how many truly belong to the positive class.

$$precision = \frac{true\ positive}{true\ positive + false\ positive}$$

Recall is defines as, out of all the items that are truly positive, how many were correctly classified as positive. Or simply we can say, how many positive items were recalled from the dataset.

$$recall = \frac{true\ positive}{true\ positive + false\ negative}$$

The traditional F-measure or balanced F-score (F_1 score) is the harmonic mean of precision and recall - multiplying the constant of 2 scales the score to 1 when both recall and precision are 1.

$$F_1 = 2 * \frac{(precision * recall)}{(precision + recall)}$$

Analysis

Data Exploration

Features are computed from a digitised image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

This database is also available through the UW CS ftp server .
Also can be found on UCI Machine Learning Repository .

Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3) Features (3 - 32)

Ten real-valued features are computed for each cell nucleus:

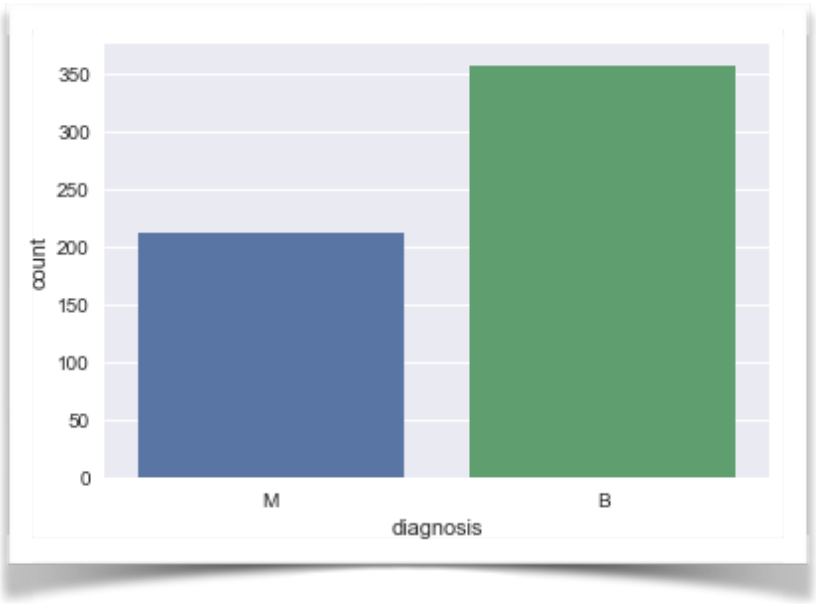
a) radius (mean of distances from centre to points on the perimeter)
b) texture (standard deviation of grey-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits.
Missing attribute values: none
Class distribution: 357 benign, 212 malignant

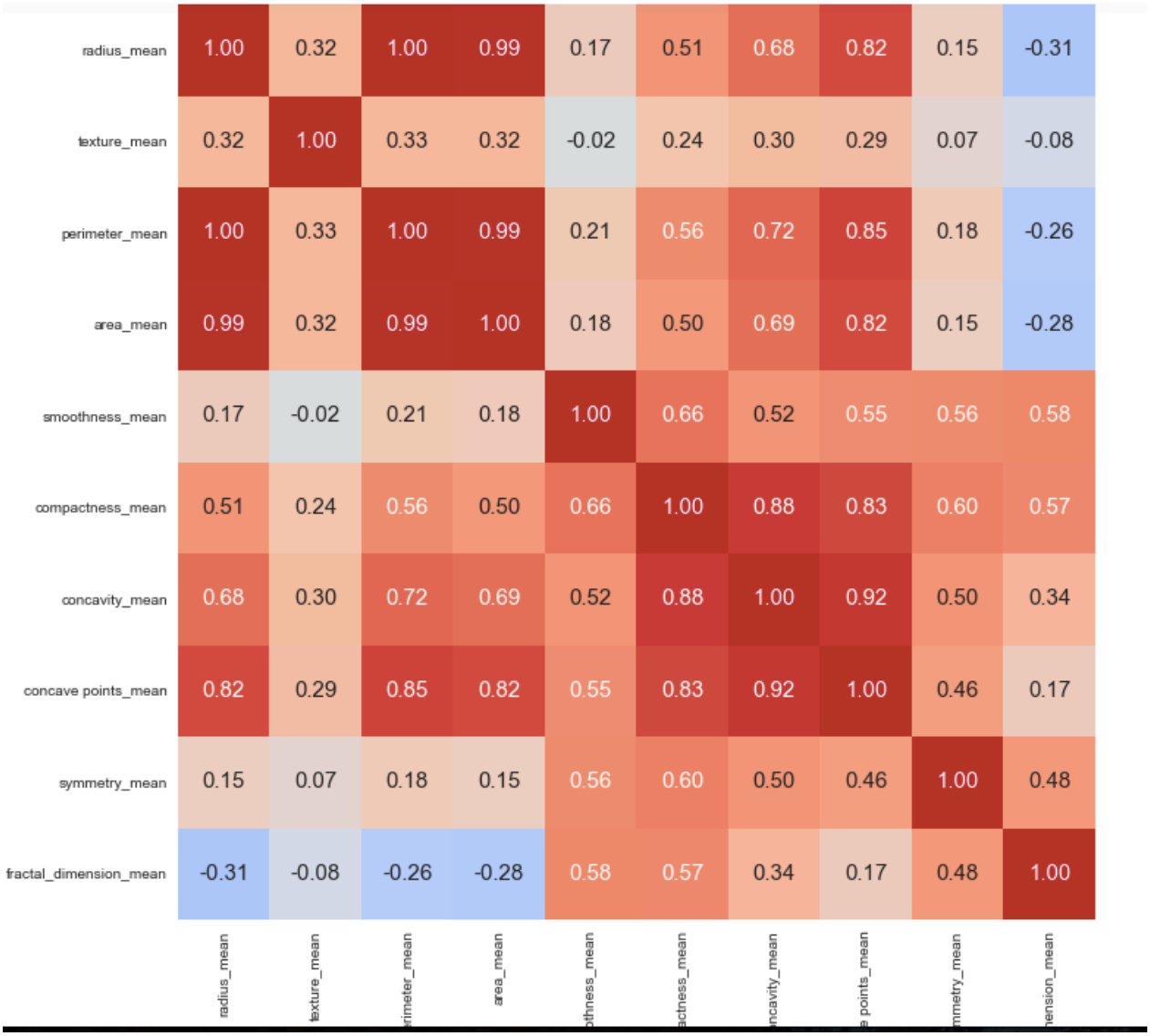
Exploratory Visualisation

The plot below shows the distribution of our dataset of patients in two classes, that are malignant and benign.



We have data of 569 patients, out of which 357 are diagnosed as benign and 212 as malignant.

One interesting thought to consider is if one (or more) of the ten real-valued features is actually relevant for understanding customer purchasing. That is to say, is it possible to determine that patients having one symptom will surely be having the other, that is to say are the features dependent on each other.



To get a better understanding of the dataset, we can construct a heat-map/correlation graph of each of the ten real-valued features (which are the mean features) present in the data. If you found that the feature you attempted to predict above is relevant for identifying a specific patient, then the correlation graph below may not show any correlation between that feature and the others. Conversely, if you believe that feature is not relevant for identifying a specific customer, the correlation graph might show a correlation between that feature and another feature in the data.

In the above graph as you can see, we are using the mean features. Actually the mean, standard error and "worst" or largest (mean of the three largest values) of all the 10 real

valued features are present, but throughout this project we will be working with the mean features.

What we observe from the above heatmap is the following,

- the radius, perimeter and area are highly correlated, so from these 3 we can choose anyone, that would solve our purpose. In our case we will be going forward with perimeter.
- the compactness, concavity and concave point seems to be highly correlated, again we have an option of picking one, hence choosing one we will be going forward with the compactness.

Algorithms and Techniques

In machine learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. In our particular problem we need to classify new patients to belong the given two classes, i.e. malignant and benign.

As we are pretty sure that it is a classical supervised classification problem, and hence we will be looking at few famous and relevant (on the basis of data set size and type) algorithms, to maximise the accuracy with which we can classify the new patients.
The five relevant supervised classification problems we will be looking at are,

Gaussian Naive Bayes: There are many real life applications of naive bayes like to name a few, to mark an email as spam or not, a text document belongs to one or other category, image classification with naive bayes, etc. Naive Bayes is a simple and easy to implement algorithm. Due to its simplicity, this algorithm might outperform more complex models when the data set isn't large enough. It is also pretty fast, e.g., if you have a large multi-class problem, you'd only have to train one classifier whereas you'd have to use One-vs-Rest or One-vs-One with in SVMs or logistic regression. Another point is that you don't have to worry so much about hyperparameter optimisation – if

you are estimating the class priors from the training set, there are actually no hyperparameters. The Naive Bayes algorithm makes an assumption that the features used are conditionally independent. This may lead to wrong decisions in case the assumptions are not met. Here i suppose the heuristics were meeting, and it is a simple case with simple categories, order and all those complex behaviours like in sentimental analysis, wont come in affect in the problem at hand. So we can give it a shot and check its outcome.

Decision Trees: Decision trees are oftenly used for determining the success of a project or marketting startegy, though complex examples also include predicting library use, Characterization of Leiomyomatous Tumors, and Star/Cosmic-Ray Classification in Hubble Space Telescope Images etc. Decision trees are simple and give direct picture of what we are trying to do, they can handle both numerical as well as categorical data, uses white box model, etc. The most important limitation of decision tree model is, it can easily create over-complex trees that do not generalise well from the training data(overfitting). Other limitations include trees do not tend to be as accurate as other approaches, trees can be very non-robust, the problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts, etc. Its kind of a simple decision making problem, that given the relevant data, to decide the output i.e. in our case is malignant or benign. I feel Decision tree would be suitable candidate for this case (unless it overfits the small amount of data), as it can predict the outcome following a decision structure formed from the fixed train data, with limited fixed column values given.

Support vector machines: SVM's are used in Protein Fold and Remote Homology Detection.Protein remote homology detection is a central problem in computational biology. Supervised learning algorithms based on support vector machines are currently one of the most effective methods for remote homology detection. There are many other applications of SVM's like SVM for Geo- and Environmental Sciences, content based image retrieval, DATA Classification using SSVM, etc. Support vector machines tend to work very well in complicated domains where there is a clear margin of separation. But do not perform well in very large datasets because the training time happens to be cubic in the size of the data set. Also they do not work well when there is lots of noise,

when the class are very overlapping you have to count the independent evidence. As in our case, it happens to be a small dataset, so the learning is feasible and it suites the properties, so we can give it a shot and check how it performs on our test set.

k-nearest neighbors: k-NN is often used in search applications where you are looking for “similar” items; that is, when your task is some form of “find items similar to this one”. You’d call this a k-NN search. The way you measure similarity is by creating a vector representation of the items, and then compare the vectors using an appropriate distance metric (like the Euclidean distance, for example). Some of the concrete examples where KNN search is applied are Concept search and Recommender systems. It can also used for k-NN classification, in the right kind of application. k-NN isn’t as efficient as a neural network or an SVM, and generally runs slower and has lower accuracy than those approaches, but it’s got some nice practical qualities. It’s easy to train (because there’s no training), easy to use, and it’s easy to understand the results. *k*-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all machine learning algorithms.As it is widely used in practice, it would be better to test it once.

Random Forest: Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Ensembles of decision trees (such as Random Forests, which is a trademarked term for one particular implementation) are very fast to train, but quite slow to create predictions once trained. More accurate ensembles require more trees, which means using the model becomes slower. In most practical situations this approach is fast enough, but there can certainly be situations where run-time performance is important and therefore other approaches would be preferred. Of course, it is important to recognise that it is a predictive modelling tool, not a descriptive tool. Overall, for fast, simple, flexible predictive modelling, ensembles of decision trees are probably the most useful single prag-

matic tool available today - but like any method they do have their limitations. Our problem scenario actually seems a very good candidate for Random forest.

Firstly we will start by quickly training our different models, each trained with the above stated algorithms, with the selected features including texture_mean, perimeter_mean, smoothness_mean, compactness_mean, symmetry_mean. We will analyse their respective F_1 scores on the training as well test set and how long does it take to train as well as test.
Then proceeding with the second step where we again will train the models with the same above algorithms each, but this time with all the ten features and with hyperparameter tuning, and using grid search which internally selects F_1 score as a scorer, and try to get the maximum accuracy as far as we can on the train as well test set.

Then we will compare all the models with all parameters, and with selected parameters, with or without hyperparameter tuning and try to get the the best model, that would give the highest accuracy rate if given a task to diagnose new patients with breast cancer.

Benchmark

I will create different models with different algorithms and different features with different respective parameter tunings, and compere the accuracy in each case. In general i want to reduce the false negatives to a sufficient level. Accuracy of at least 94% on the test set would be acceptable for the real world cancer diagnosis. So i will tune my model in such a way to receive at minimum an F_1 score of 0.94 on the test set.

Methodology

Data Preprocessing

The preprocessing steps performed in the notebook are,

- We don't want the id column in our analysis, so we will sort of remove it.
- We will also remove the unnamed: 32, column from our data set, as it is of no use.
- For each dataset mean, standard error and "worst" or largest (mean of the three largest values) of all the 10 real valued features are present, but throughout this project we will be working with the mean features.
- Diagnosis column in the data is a non-numeric column that needs to be converted! It is simply M/B. This can be reasonably converted into `1/0` (binary) values.
- Separate the patients data into feature and target columns.
- Feature selection, to remove the highly correlated features, done using the correlation graph(heatmat). (refer to Exploratory Visualisation section). The new selected features are,
  - texture_mean
  - perimeter_mean
  - smoothness_mean
  - compactness_mean
  - symmetry_mean
  - fractal_dimension_mean

- Shuffle and split the data (both features and corresponding labels) into the number of training and testing points, with around 25% records lying in the test set. This results in 425 training points and 144 test points. We use a random state of 10 in the split function.

Implementation

Firstly we will create three helper functions which we will use for training and testing the five supervised learning models chosen above.

The functions are as follows:

- `train_classifier` - takes as input a classifier and training data and fits he classifier to the data.
- `predict_labels` - takes as input a fit classifier, features, and a target labeling and makes predictions using the F_1 score.

- `train_predict` - takes as input a classifier, and the training and testing data, and performs `train_clasifier` and `predict_labels`.
  - This function will report the F_1 score for both the training and testing data separately.

We will now import the five supervised learning models as stated above and run the `train_predict` function for each one.

- We will initialise the models and store them in respective variables.
- Fit each model with training set containing only the five uncorrelated features and make predictions on the test set which also contains just those five features.

Refinement

We will now fine tune the models which allow for, we will use grid search with some important parameters tuned and try to get the maximum F_1 score.

Firstly we will create a helper function namely, train_predict_gridsearch which will accept the model, parameters, training and testing data, and will determine the F_1 score for both train and test data. We will be using F_1 scorer with the pos_label set to 1.

Resultantly for each of our models we will,

- Create a dictionary of parameters we wish to tune for the respective model.
- Initialise the classifier and store it in a variable
- Perform grid search on the respective classifier using `f1_scorer` as the scoring method.
- Fit the grid search object to the training data, which contains all the 10 real valued features.

We will go through the parameters we tried for each of the algorithms, as we are using grid_search so it will automatically detect the best parameters from the options provided.

- Gaussian Naive Bayes: There are no real parameters we can tune in this case. We can check this by running estimator.get_params().keys(), which gives empty list in case gaussian naive bayes. So we will stick with the same result as we got earlier.

- Decision Trees: Here we tried three parameter tuning which includes,
  - max_features with possible values as auto, sort, log2
  - min_samples_split with possible values from 1 to 10
  - min_samples_leaf with possible values from 1 to 10

- SVM: Here again we tried three parameters for tuning which include,
  - kernel with possible values as linear or rbf
  - C with possible values from 1 to 10
  - gamma with 4 possible values 0.001, 0.01, 0.1, 1

- KNN: Here also we will be trying three different parameters which include,
  - n_neighbors with possible values from 1 to 10
  - leaf_size with possible values from 1 to 10
  - weights with possible values as uniform and distance

- Random Forest: In this particular case, firstly we try to train the model with the complex 10 real valued features and check the predictions on the test set. Then we try to get the feature importance, with the help of this model.

perimeter_mean	0.32214
area_mean	0.249135
concavity_mean	0.109236
radius_mean	0.092945
concave points_mean	0.074797
texture_mean	0.072671
compactness_mean	0.038808
smoothness_mean	0.016588

fractal_dimension_mean	0.014939
symmetry_mean	0.008741

So with the help of above knowledge gained, we try to use the most important features which are as,

- concave points_mean
- area_mean
- perimeter_mean
- radius_mean
- concavity_mean

Then we again trained the new random_forest model and make predictions using train_predict function defined earlier.

But we observe the performance on the test set, with the model using all the 10 real valued functions, was the best. Hence we tried to tune this model using the parameters,

- max_depth with possible values of 3 and none.
- max_features with possible values of 1, 3 and 10.
- min_features_split with possible values of 1, 3 and 10.
- min_samples_leaf with possible values of 1, 3 and 10.
- bootstrap with possibilities on either end, true or false.
- criterion with probable values as gini or entropy.

Results

Model Evaluation and Validation

During the process we analyse the train time , testing time, F_1 scores of both training and testing. The F_1 score of test set was given the most importance.
The result we achieved from the first phase of implementation model, that is training on the models without any hyper parameters tuned are,

Classifier	Training Time	Prediction Time (test)	F_1 Score(train)	F_1 Score(test)
GaussianNB	0.0091	0.0006	0.8758	0.9216
DecisionTree	0.0036	0.0005	1.0000	0.8909
SVM	0.0218	0.0022	0.9000	0.8381
KNN	0.0011	0.0010	0.9032	0.8491
RandomForest	0.0472	0.0025	0.9905	0.9216

Hence we observe from the upper models and their respective scores that Random Forest model tends to outperform other models, in terms of their F_1 score on the test set. Which is what we assumed in the starting.

Hence we tried some sort of parameter tuning in each of the models(apart from Gaussian NB which doesn't permits) and check to see if the results remain the same in terms of comparison with others, and how well we can tune our final model to maximise the accuracy with which it can diagnose the cancer.
So the resultant tuned parameter result which we achieve is as follows,

Classifier	Prediction Time (train)	Prediction Time (test)	F_1 Score(train)	F_1 Score(test)
GaussianNB	0.0007	0.0007	0.8758	0.9216
DecisionTree	0.0001	0.0005	0.9350	0.9216
SVM	0.0009	0.0005	0.9038	0.8679
KNN	0.0040	0.0021	0.8472	0.8125
RandomForest	0.0011	0.0012	0.9554	0.9515

As we analyse the above table, it is very much obvious that our Random Forest model with properly tuned parameters perform the best in terms of F_1 score on training as well as test set.
The best parameters for tuning our final model i.e. Random Forest are,

bootstrap	TRUE
min_samples_leaf	3
min_samples_split	10
criterion	gini
max_features	1
max_depth	None

The final F_1 score which we achieved for training is 0.9554 and for test is 0.9515.

Justification

Earlier in the Benchmark section, we stated that minimum we needed 94% accuracy on the test set, for proper and reliable diagnosis of breast cancer.
Using the Random Forest ensemble learning model with proper tuning, we were able to achieve 95.15% accuracy on the test set. That means our purpose of reducing the false negatives is reduced by a lot, and our model will be able to make efficient predictions.

Still it is around 95% correct, not the 100%, but still for our purpose it is best, as human's eye or doctors were not able to achieve this much accuracy in diagnosis of breast cancer, and it often leads to wrong medication. Our model improves in the accuracy aspect, trying to judge the type more accurately.

Conclusion

Free Form Visualisation

We took an approach of going through all the examples and analysed the performance and the tuning of 5 of the relevant algorithms with some of the hyperparameters.
The best parameters for each of the algorithms are,

Decision Tree	max_features: log2, min_samples_split: 4, min_samples_leaf: 10
SVM	kernel: linear, C: 4, gamma: 0.001
KNN	n_neighbors: 10, weights: uniform, leaf_size: 1
Random Forest	bootstrap: True, min_samples_leaf: 3, min_samples_split: 10, criterion: gini, max_features: 1, max_depth: None

The random forest is best suited for predictive tasks , and intact the ensembles of decision trees the most useful single pragmatic tool available today, and hence it proved its worth.

Reflection

The process used for this project can be summarised using the following steps:

I. An initial problem and the relevant public datasets were found.
II. The dataset was downloaded and analysed.
III. The dataset was processed to make it ready for analysis.
IV. A correlation graph is created and uncorrelated features are identified.
V. A benchmark was created for the classifier.
VI. Dataset was partitioned in the train and test set.
VII. Relevant supervised learning algorithms were identified.
VIII. The algorithms were tuned and grid search was used for analysis.
IX. The analysis results were compared against all the tuned models.

I found the 8th step was a little time consuming step, as we have to resolve for trial and error to get the exact parameters. Plus in step 7th we need to find the relevant algorithms, where scikit learn's page proved very helpful.
It took some time even to analyse the dataset in step 2 as it took some time to identify and get familiarise with breast cancer symptoms and diagnosis.

Its a very interesting problem for me to solve, as i have seen many people facing severe adverse health affects or even casualties just because of wrong diagnosis. And coming to breast cancer which is the most common form of cancer now-a-days, so this was my first choice when i started my machine learning journey.

The result of around 95% is very satisfactory, as present diagnosis trends suffer a lot of wrong diagnosis. And the worst of all is the false negatives and even false positives for psychological reasons. So our result is satisfactory for the starting with the amount of data we manage to find.

Improvement

Firstly in this section i would make a point about the data. If more amount of real data was available in front of us, the results would be much better than the present one. I the dataset was large we could have gone for K-fold cross validation dividing our dataset in 3 sets i.e. train, test and validation set, instead of 2.

But i think this is the right section for making this point, i think if we could arrange the real images of the specimen or the piece of the infected cells, and we can get the exact picture of the cell, we could have used neural networks, specially the convolutional neural nets to get the better results, because i fell the specimens guarantee we are taking above in the form of area and others could also be faulty and hence will make our model perform badly in real world situation. If we could get the picture of the datasets we could use say leNet model, (with some modification, maybe make it two way, etc.), and help give better results. We definitely could be pretty sure about those results.

Though in the present scenario with the fine needle aspiration way, and the present data records what we have achieved is satisfactory for the starting point, as this much accuracy definitely paves our way for future enhancements in the field.