Capstone Project Ebrahim Jakoet

Machine Learning Engineer Nanodegree Aug 22nd, 2019

# Definition

## Project Overview

About 12% of women living in America will develop some form of invasive breast cancer. It is estimated that 41,760 of women in the US will die from breast cancer in 2019. Breast cancer remains among the most commonly diagnosed types of cancer in older women especially, and a leading cause of death compared to most other types of cancer [1]. In this project we will use the Breast Cancer Wisconsin (Diagnostic) Data Set to build a model that can predict the presence of a malignant cancer based on the given features in the dataset. We will also determine which features are more significant in the detection of malignant breast cancer. The dataset is derived from digitized images of a fine needle aspirate (FNA) biopsy of breast mass samples. The features describe characteristics of the cell nuclei present in the image [^2]. Machine learning techniques have already been used in the diagnosis of breast cancer from Fine Needle Aspirates [3][4]. Early detection of breast cancer can be life-saving and Machine Learning is increasingly being relied upon in early detection methods. It is the real life-saving results using machine learning that draws me to the subject.

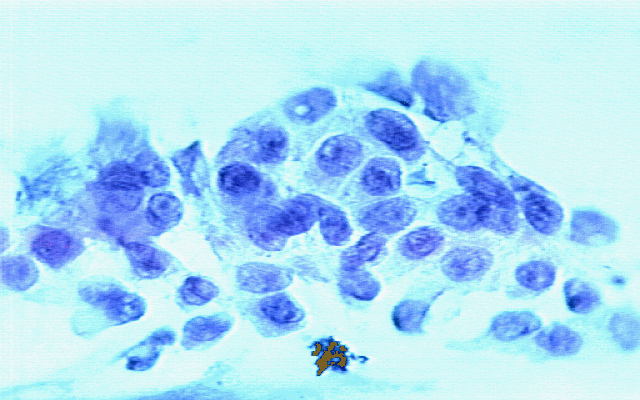


Figure 1. A sample image of an FNA biopsy.

## Problem Statement

There are 30 feature measurements contained in the dataset with each sample labeled as either (M) Malignant or (B) Benign. The goal is to determine which features are most important and to build and compare an AdaBoost model and a DNN model that will be able to predict the malignancy of the FNA sample based on the given features. The 2 models will be compared using standard evaluation metrics.

The first model will be an AdaBoost ensemble model and will be used as the benchmark model. The second model will be a Deep Neural Network (DNN) model built with the Keras modules for neural network architecture. AdaBoost and neural network algorithms are popular algorithms in machine learning and have been compared in a number of studies before [5][6]. The 2 competing models will be compared by their evaluation metrics as described below. We will also identify the most important features in the feature set.

## Metrics

To compare the AdaBoost and DNN models, we will be using an accuracy score, recall and processing time. Precision and F1 score will also be calculated, but not considered as important for our comparison. Accuracy will give us a measure of how well the model is able to predict the correct outcome. Recall is important because we want to be able to minimize the False Negatives since this is a test for the presence of cancer in a breast mass. Processing time is only relevant if the deployment of the solution has limited resources. It is always good to see how the 2 models compare in time. For processing time, we will use the python time modules to calculate time difference between the training start and stop of each of the 2 models. The formulae for the accuracy, precision, recall, F1 Score and processing time evaluation metrics are shown below.

*Accuracy = True Positives + True Negatives / Total number of Samples*

*Precision = True Positives / (True Positives + False Positives)*

*Recall = True Positives / (False Negatives + True Positives)*

*F1 = 2 \* (precision \* recall) / (precision + recall)*

*Processing Time = Time elapsed between (Training start, Prediction end)*

# Analysis

## Data Exploration

The Breast Cancer Wisconsin (Diagnostic) Data Set was acquired from the Kaggle website at <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data>. It is also available from the UCI Website Archive at <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29>. And from the ftp server at <ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/cancer/>. The data consists of a single file of interest of 32 columns and 569 sample instances. Creators of the data set are listed below:

* Dr. William H. Wolberg, General Surgery Dept., University of Wisconsin, Clinical Sciences Center, Madison, WI 53792, wolberg@eagle.surgery.wisc.edu
* W. Nick Street, Computer Sciences Dept., University of Wisconsin, 1210 West Dayton St., Madison, WI 53706,street@cs.wisc.edu 608-262-6619
* Olvi L. Mangasarian, Computer Sciences Dept., University of Wisconsin, 1210 West Dayton St., Madison, WI 53706 olvi@cs.wisc.edu

The data consists of an ID column, a labeled target column that shows (M) for malignant or (B) for Benign and the following 10 features.

1. radius (mean of distances from center to points on the perimeter)

2. texture (standard deviation of gray-scale values)

3. perimeter

4. area

5. smoothness (local variation in radius lengths)

6. compactness (perimeter^2 / area - 1.0)

7. concavity (severity of concave portions of the contour)

8. concave points (number of concave portions of the contour)

9. symmetry

10. fractal dimension ("coastline approximation" - 1)

Each of these 10 features have a mean, standard error and worst or largest (mean of the three largest values) value which makes the total feature set 3 x 10 = 30 features in total. Feature values have been recorded with 4 significant digits.

A sample of the raw data is shown in the Table below.

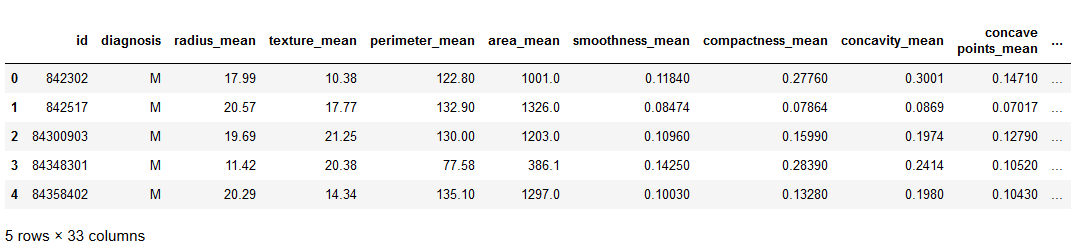


Table 1. A sample of the raw data taken FNA biopsy images.

There are 2 columns in the raw data, viz. **id** and **Unnamed: 32** that were dropped since they carry no significant information for our task. We encoded the diagnosis column with **1** and **0** representing Malignant and Benign respectively. After the 2 columns were dropped, there were no missing or problematic feature values left in the data set. The counts of Malignant and Benign samples in the data is shown below.



Figure 2. Count of Malignant and Benign diagnoses in the data.

A sample of a table with the statistics of the data is shown below.

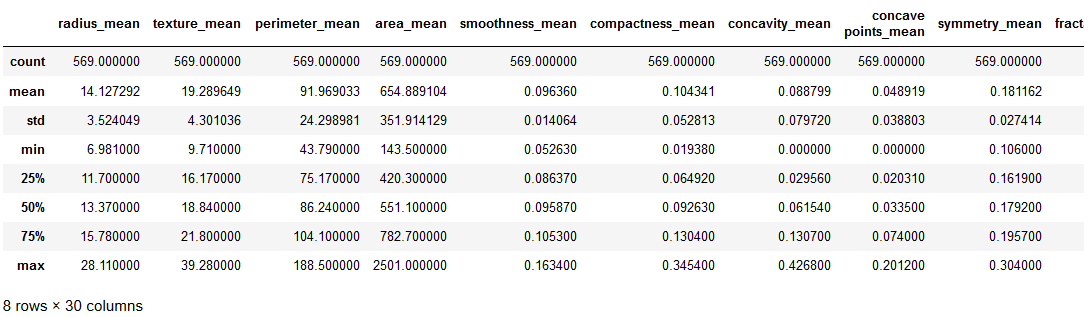


Table 2. Description of basic statistics in the feature set.

We notice that the feature ranges and means are significantly different which we will have to normalize later for a Deep Neural Network model. Also, since we have the mean values, standard error (se) and the mean of the top 3 worst values for each significant feature, we expect that the feature set may be reduced by removing some features that are highly correlated or not as important.

## Exploratory Visualization

Since the feature set consist of 30 columns, we are particularly interested to see if there are highly correlated features that we can use to reduce the input size for the models that we will use. The pair-plot below is a graphical representation of how each of the features are related to every other feature in the feature set.

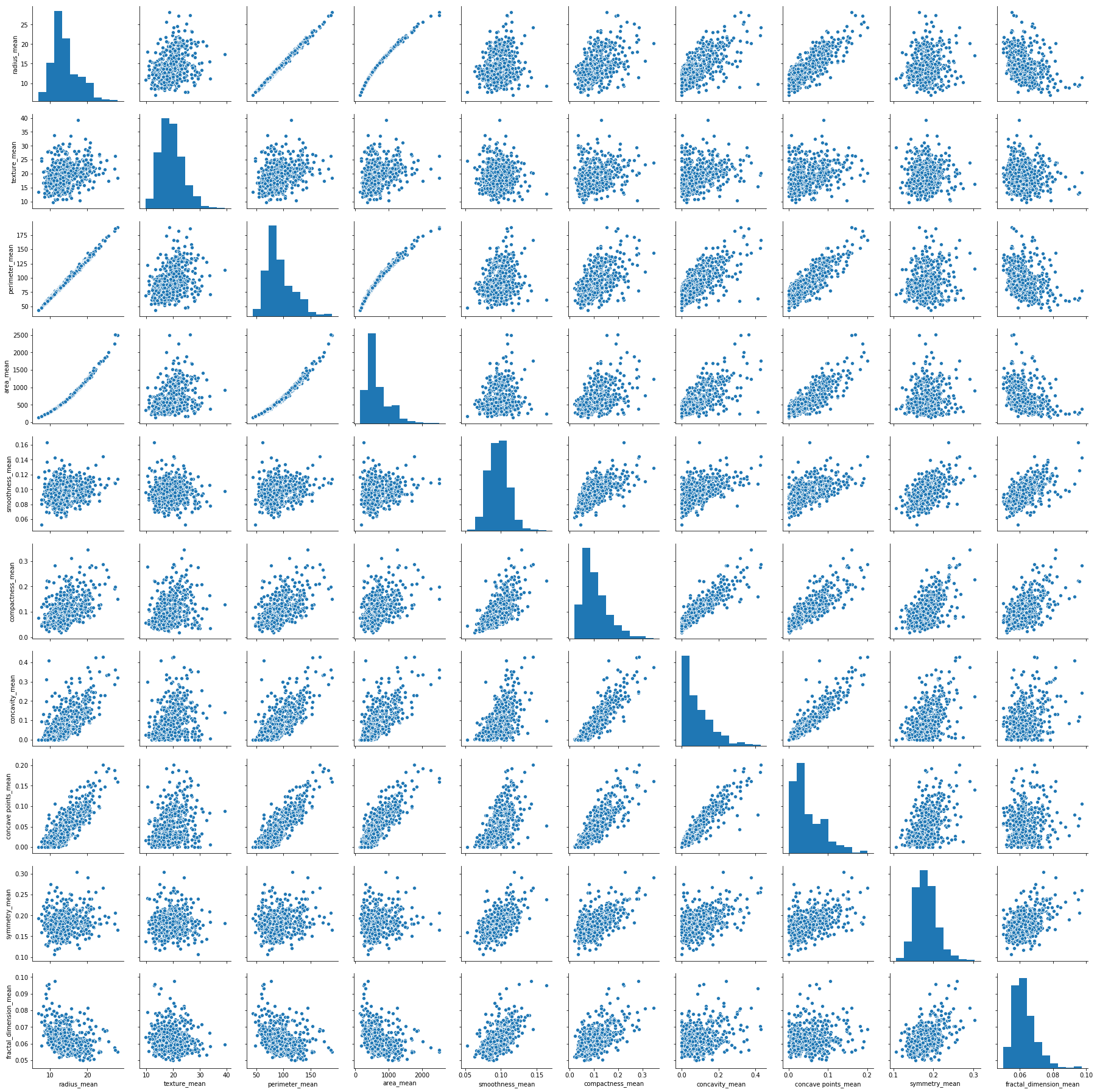


Figure 3: Pair-plot of features showing the pair-wise relationship between features.

We notice that the radius\_ features are highly correlated with the area\_ and perimeter\_ features. This is to be expected given the clearly defined relationship between radius, area and perimeter (or circumference). We also notice that the concavity\_ features are correlated to the concavity\_points features. This relationship is confirmed by looking at the Pearson’s correlation matrix below.

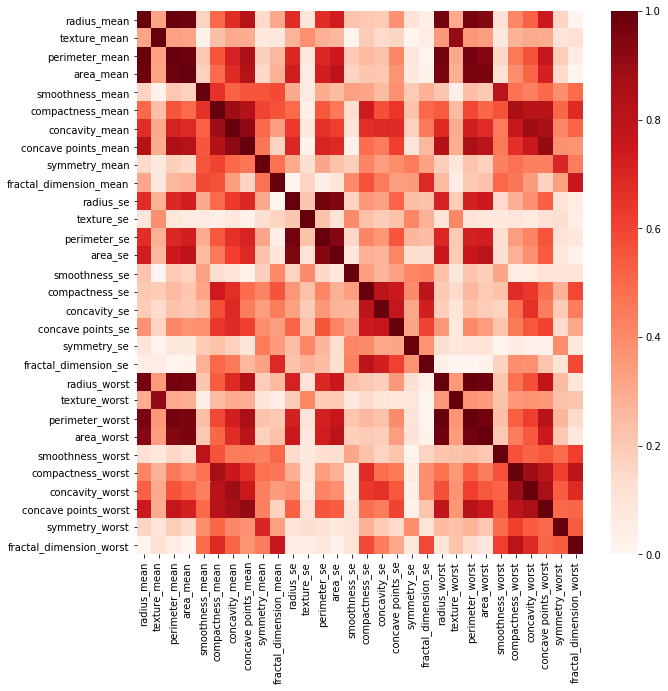


Figure 4: Pearson’s Correlation Matrix

We used joint-plots to have a closer look at the relationship between radius, area and perimeter means, as well as concavity\_mean and concavepoints\_mean.

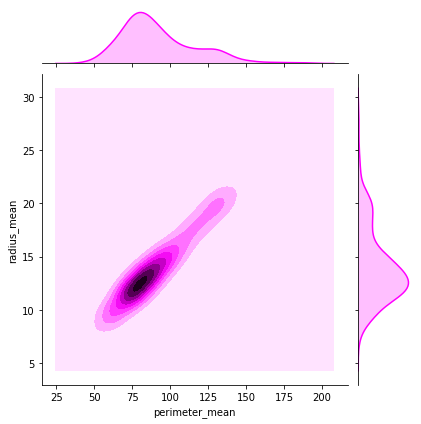


Figure 5: Joint plot of radius\_mean and perimeter mean features.

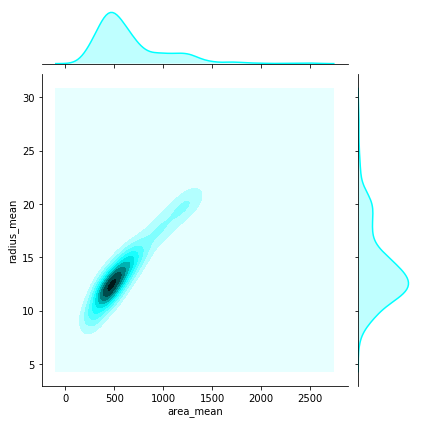


Figure 6: Joint plot of radius\_mean and area\_mean features.

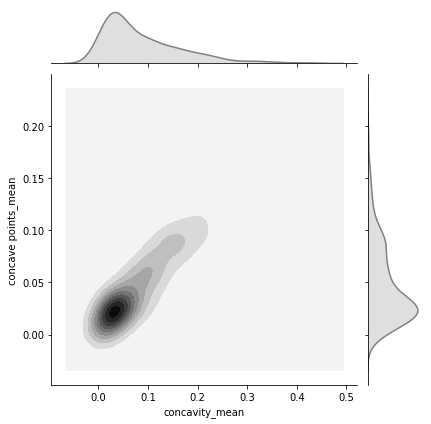
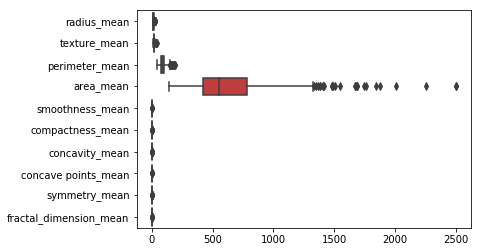
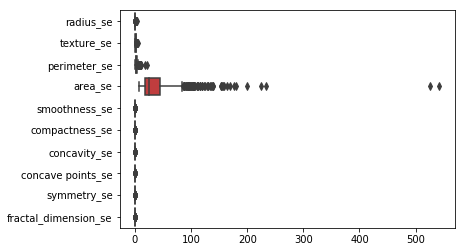


Figure 7: Joint plot of concavity\_mean and concavity points\_mean features.

From this we conclude that it is not necessary to include the radius\_, area\_ and perimeter\_ features and instead we will only include the radius\_ values. For the concavity\_ and concave points\_ features, we will retain only the concavity\_ features.

Another interesting characteristic of the feature set is the distribution of points in each feature category. We used boxed plots to illustrate how the feature distributions compare to each other. The features were separated according to their aggregation categories (mean, se, worst) and then illustrated in a box-plot.





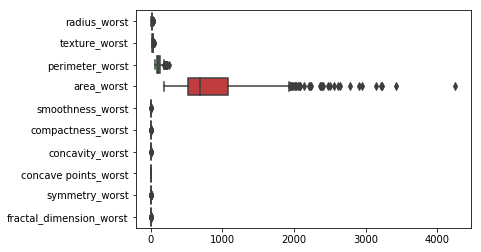


Figure 8: Distributions of Features per Aggregation Category

## Algorithms and Techniques

After filtering and preprocessing the data, we will employ 2 classification models, the first using AdaBoost and the second a Deep Neural network. AdaBoost and neural network algorithms are popular algorithms in machine learning and have been compared in a number of studies before [5][6]. Both these models have hyper-parameters that can be tweaked to refine the model to achieve the highest possible evaluation metric scores. Each of the models and their hyper-parameters are described below.

### AdaBoost

We used an AdaBoost Model as our benchmark model. AdaBoost is a boosting method that uses a number of weak classifiers combined through a weighted voting scheme that is trained to produce the final result. The following Hyper-parameters can be refined to improve the model performance:

## 1. base\_estimator: DecisionTreeClassifier(max\_depth=1)

## 2. n\_estimators: 1000

## 3. learning\_rate: 1.0

## 4. algorithm: SAMME.R

## 5. random\_state: 19

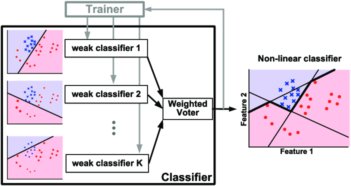


Figure 9. Illustration of AdaBoost Model.

### Deep Neural Network Model

The competing model will be a Deep Neural Network. We will build this model with Keras. The model architecture will be the input layer consisting of 23 features, 2 hidden layers of 16 nodes each with *relu* activation functions and a dense layer for the output with *softmax* activation function. The DNN model will use the *binary\_crossentropy* loss function with the *rmsprop* optimizer against the `accuracy` metric. We will train the model using *batch\_size* of 32 for 1000 epochs.

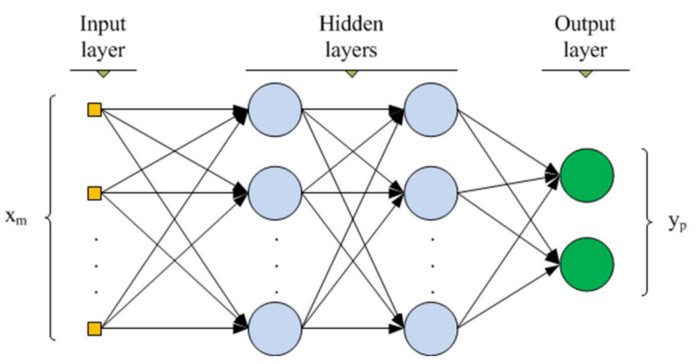


Figure 10. Illustration of DNN with 2 hidden layers.

## Benchmark Model

## In our case, since we compared 2 competing models, the AdaBoost model (already described above) was used as the benchmark model. The evaluation metrics of accuracy and recall, in particular, but also precision, F1 Score and processing time was used for comparison with our Deep neural network model. The model was also used to determine the most important features from the feature set. The learning rate was refined to produce the best evaluation metrics to be compared against. We also used a defined random\_state to improve the possibility of getting improved results on successive attempts of running the model.

# Methodology

### Data Preprocessing

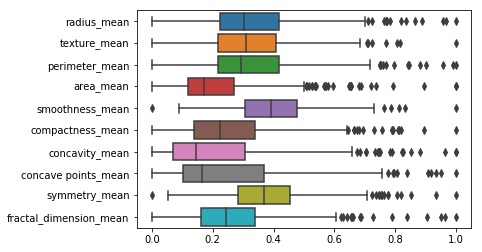
Since the original data file is in a compressed format, we decompressed the data file and then read it into a Pandas data frame, using the variable *X* for the features and *y* for the labels.

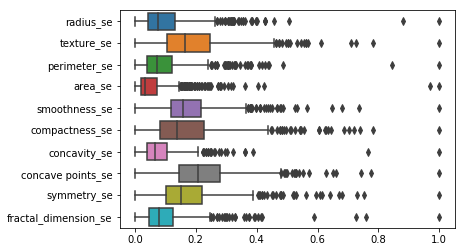
We explored the possibility of using the Low Variance method to reduce the number of features to be used for the AdaBoost and DNN models. We used a Low Variance threshold value of *0.9 x (1 - 0.9)*. However, since we used a normalized feature set for the DNN, but not for the AdaBoost model, this method of reducing the feature set was not suitable.

Based on the high correlation seen between radius\_, area\_ and perimeter\_ feature values, as well as between concavity\_ and concave points\_ feature values, the following features were dropped, thus reducing the feature set from 30 features to only 23 features.

* area\_mean
* perimeter\_mean
* area\_worst
* perimeter\_worst
* area\_se
* perimeter\_se
* concave points\_mean

Since it is clear from the box-plots that the ranges, means and variances differ considerably for the different features, with the numerical values of area being larger than the rest, we chose to normalize the feature set for the Deep Neural network model. Normalized distributions of this data are shown below.





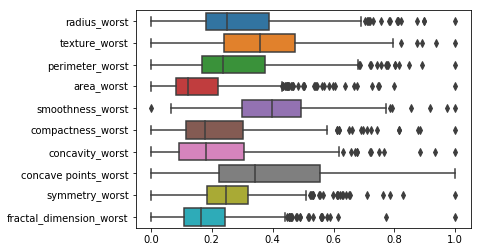


Figure 11: Normalized Features per Feature Aggregation Category.

### Implementation

The following flowchart and discussion that follows describes the process followed for our model comparison.

#### Data Validation and Exploration

We completed some basic data exploration by checking for valid data, removing invalid on unnecessary data, generating some statistics on the features and plotting charts to show some of the characteristics of features. We used the `seaborn` Statistical Data Validation toolset for all visualizations.

#### Feature Selection and Normalization

We investigated the features, using the low variance method to see if there were any features with low variance that could be excluded. We used a Low Variance threshold value of *0.9 x (1 - 0.9)*. However, this method proved unsuitable for the model comparison since we used normalized features using the MinMaxScaler for the DNN only. We also used the pearson Correlation Matrix to identify highly correlated features which were removed as described before.

#### Split into Training and Test datasets

Using the *sklearn.model\_selection.train\_test\_split()* module, we split the data into a training set and a testing set. This was done for both the feature set after filtering with and without normalization. We used 20% of the data for testing, 80% for training, with the `shuffle` parameter set to True to ensure a good mix of malignant and benign samples in both training and testing data sets.

#### Build the Benchmark Adaboost Model

We built an AdaBoost Model as our benchmark model. The following hyper-parameters were used as a baseline, but learning rate was later refined to 0.9 to produce the best evaluation metric results for the benchmark model.

## 1. base\_estimator: DecisionTreeClassifier(max\_depth=1)

## 2. n\_estimators: 1000

## 3. learning\_rate: 1.0 (changed to 0.9 on refinement)

## 4. algorithm: SAMME.R

## 5. random\_state: 19

After fitting the model to the training set, we used it to predict the outcome on the test data set. Thereafter we calculated the following evaluation metrics:

*Accuracy = True Positives + True Negatives / Total number of Samples*

*Recall = True Positives / (False Negatives + True Positives)*

*Processing Time = Time elapsed between (Training start, Prediction end)*

#### Build the DNN Model

The competing model was a Deep Neural Network. We built this model with Keras. The model architecture consisted of the input layer, 2 hidden layers of 16 nodes each with *relu* activation functions and a dense layer for the output with *softmax* activation function. The DNN model used the *binary\_crossentropy* loss function with the *rmsprop* optimizer against the `accuracy` metric. We trained the model using *batch\_size* of 32 for 1000 epochs as a baseline model. Thereafter the batch size was changed to 16 and 64. Since the best result was found using batch\_size of 64, we used that batch\_size to further change the model by adding drop\_out layers with a drop\_out value tweaked to 0.4 for the best result.

### Compare Evaluation Metrics

Once both models were trained and used to predict the outcomes on the test set, we compared the results of the evaluation metrics for each of the models. Finally we will drew some conclusions on the strengths and weaknesses of the competing models based on these results.

## Refinement

The AdaBoost model and Deep Neural Network model were both refined with multiple runs adjusting the hyperparameters. For the AdaBoost model the final Hyperparameter values selected are shown below:

## 1. base\_estimator: DecisionTreeClassifier(max\_depth=1)

## 2. n\_estimators: 1000

## 3. learning\_rate: 0.9

## 4. algorithm: SAMME.R

## 5. random\_state: 19

The initial results with learning\_rate set to 1 is shown below:

|  |  |
| --- | --- |
| Accuracy score | 0.947 |
| Precision score | 0.976 |
| Recall score | 0.889 |
| F1 score | 0.930 |
| Elapsed Time | 1.531 |

Table 3: Evaluation Metrics for First AdaBoost Model

|  |  |
| --- | --- |
| Accuracy score | 0.956 |
| Precision score | 1.000 |
| Recall score | 0.889 |
| F1 score | 0.941 |
| Elapsed Time | 1.547 |

Table 4: Evaluation Metrics for Final AdaBoost Model

In the final AdaBoost model, the Accuracy score, Precision score and F1 score is higher with the processing time being just slightly more than the previous model.For the DNN, the baseline model had 2 hidden layers of 16 nodes with relu activation functions, using a batch\_size of 32 and no Dropout layers. Below are the results for the baseline DNN model.

|  |  |
| --- | --- |
| Accuracy score | 0.974 |
| Precision score | 0.957 |
| Recall score | 0.978 |
| F1 score | 0.967 |
| Elapsed Time | 47.938 |

Table 5: Evaluation Metrics for Baseline DNN Model

After refining the baseline model by changing the batch\_size and experimenting with adding dropout layers, the model that produced the best results had batch\_size of 64 with no dropout layers. The evaluation metrics for the best DNN model is shown below.

|  |  |
| --- | --- |
| Accuracy score | 0.982 |
| Precision score | 1.000 |
| Recall score | 0.956 |
| F1 score | 0.977 |
| Elapsed Time | 27.094 |

Table 6: Evaluation Metrics for Best DNN Model

# Results

There were 2 goals to this project viz. find the most important features in the data set and find the best of the 2 competing models between AdaBoost and a DNN. We used the results from the AdaBoost model to find the most important features. This is shown in the chart below:

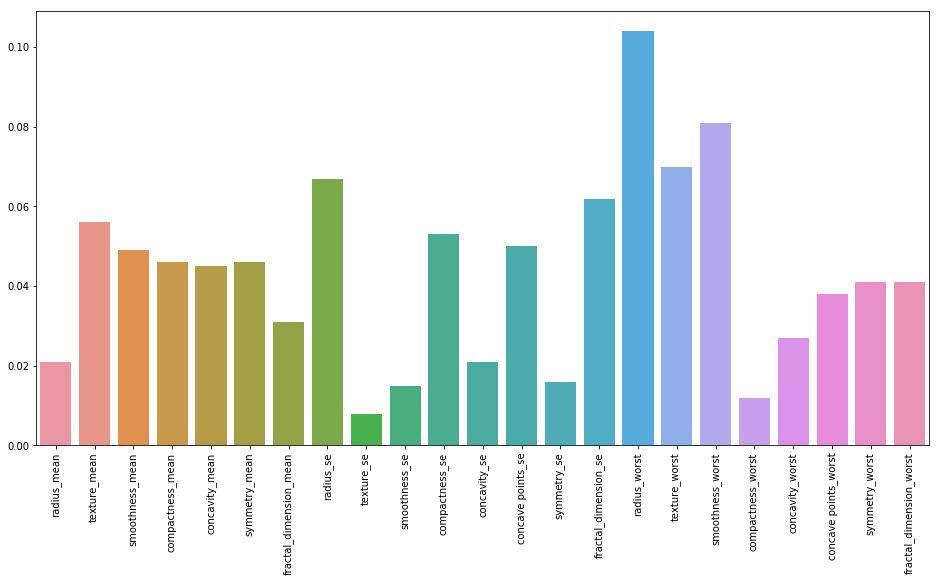


Figure 12: The Most Important Features

The table below shows the results of all the models built for each of the evaluation metrics.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Evaluation Metric | AdaBoost, lr=1.0 | AdaBoost  lr =0.9 | DNN, bs=32, no Dropouts | DNN, bs=16, no Dropouts | DNN, bs=64, no Dropouts | DNN,  bs=64, with Dropout=0.4 |
| Accuracy score | 0.947 | 0.956 | 0.974 | 0.974 | **0.982** | 0.965 |
| Precision score | 0.976 | 1.0 | 0.957 | 1.0 | **1.0** | 0.977 |
| Recall score | 0.889 | 0.889 | 0.9778 | 0.9334 | **0.9556** | 0.933 |
| F1 score | 0.930 | 0.941 | 0.967 | 0.966 | **0.977** | 0.955 |
| Elapsed Time | 1.531 | 1.547 | 47.938 | 90.734 | **27.094** | 37.797 |

Table 7: Evaluation Metrics of all AdaBoost and DNN models

## Model Evaluation and Validation

The final model selected from all the models tested is the Deep Neural Network with 2 hidden layers of 16 nodes each, with relu activation functions and a Dense output layer with sigmoid activation function. The DNN architecture does not have Dropout layers and the batch size = 64.

This model was selected based on the results shown in Table 7 above. This model has the highest accuracy, precision and F1 scores with the second highest recall score. Only the DNN with batch-size of 32 and not Dropouts had a recall score higher than this model. Even though recall is important to us for a Breast Cancer Malignancy test, it is a small trade-off for the gains on all other metrics.

Several runs of this model have shown it to be superior. The input has been varied by the shuffle=True setting in the splitting of the training and testing data sets. Dropout layers have not improved this model either.

## Justification

I believe the final solution is the best solution of all the models tested. While the DNN takes significantly longer to train, it has produced higher evaluation metrics in every evaluation category. The table below shows the Benchmark AdaBoost model compared to the Final DNN Solution.

|  |  |  |
| --- | --- | --- |
| Evaluation Metric | Benchmark Model  AdaBoost  lr =0.9 | Final Solution  DNN, bs=64,  no Dropouts |
| Accuracy score | 0.956 | **0.982** |
| Precision score | 1.0 | **1.0** |
| Recall score | 0.889 | **0.956** |
| F1 score | 0.941 | **0.977** |
| Elapsed Time | 1.547 | **27.094** |

Table 8. Comparing the Final Solution to the Baseline Model

# Conclusion

## Free Form Visualization

One significant conclusion from this project is in regard to the features found to be most important. Based on the most important features shown in Figure 8 above, we notice that the radius\_worst, smoothness\_worst and texture\_worst , in that order, are the most important features for this data set. This is also intuitively what we would expect, however, I was expecting the mean radius to be among the top 3 most important features as well.

What is even more interesting is that when we look at the relational plots of radius\_worst with the smoothness\_worst and texture\_worst, we notice that malignant samples are very closely tied to the radius\_worst feature. In fact, where the radius\_worst is mostly above a value of 15 – 20, the malignancy outcome is very likely and above 25 virtually a certainty.

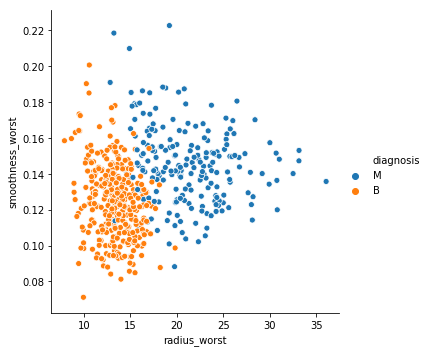


Figure 9: Relational plot of diagnosis for radius\_worst vs smoothness\_worst

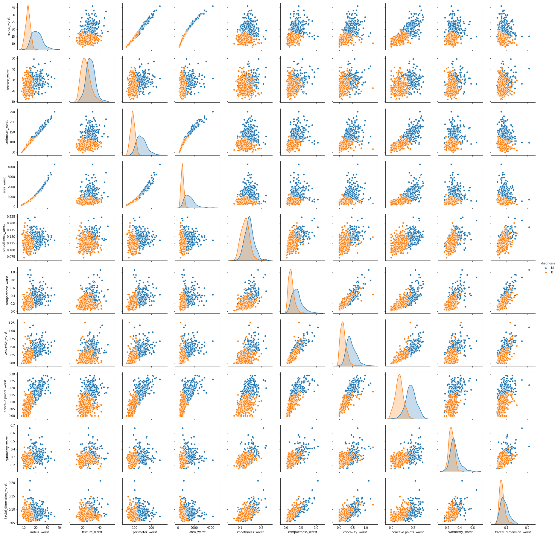


Figure 10: Pair-plot showing relationship between all worst features.

Looking at the pair-plot of all the worst features we notice distinct peaks a distance from each other in the distributions on the diagnonal for radius\_worst and concave\_points\_worst. The relational plot of these 2 features show that they may be used wit a single line discriminator for the malignancy test.

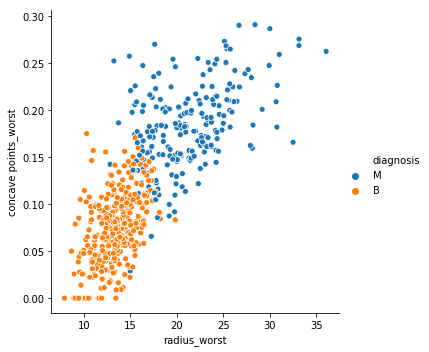


Figure 11: Relational plot of diagnosis for radius\_worst vs concave points\_worst

Another interesting conclusion drawn from the results is that, for this project, the Deep Neural Network results from the best tuned model repeatedly produced better results compared to the best tuned AdaBoost model. However, the DNN processing time was significantly larger than that required for the AdaBoost model. This could be a consideration where a trade-off between processing time and accuracy needs to be made for the deployed solution.

## Reflection

This was a very interesting project to work on and I learnt a great deal about breast cancer and the machine learning algorithms we used for classification. Building the process from data exploration to feature selection to building and comparing the models gave me a good view of the process end-to-end. There were several challenges in working with the data and trying to find the best fit models for the solution.

One of the greatest challenges was in deciding on hyper-parameters for the 2 models. For the DNN, I had to research guidelines for selecting the number of hidden layers and the number of nodes per hidden layer. Then selecting the batch size and number of epochs also appears to be rather arbitrary at first and you really need to get a feel for what seems to work well. This appears to be more of an art rather than a science, but I was pleased with the results of the architecture and hyper-parameters I selected.

Feature selection seems to be key in fine-tuning the models to produce the best results. Initially I used the full feature set from the raw files and was not able to get as good results for either the AdaBoost or DNN models. I had a failed attempt in trying to reduce the feature set with the Low Variance Feature Selection method. Using the Pearson’s Correlation Matrix was the best approach I found for this project. I believe the final solution using the DNN can be a practical solution for this classification problem.

Building insights into the most important features and how they relate to each other was really interesting for me. For future research, it would be interesting to see if we could get better or worse classification by using only the top 3-5 features or possibly even 2-3 features. Certainly, the radius\_worst feature is a dominant feature for determining the malignancy of the breast mass. Having this knowledge alone could be very useful for future diagnoses.

## Improvement

Based on my analysis, it appears that the feature set can be further reduced. It may be possible to select anywhere from only 2 – 5 features to build a practical model without too much loss of accuracy. We could then use a clustering algorithm like the k-means algorithm to draw conclusions from how these features are clustered. This is one area that can be further researched.

Also, for the DNN, it is possible to explore more hidden layers and/or a different number of nodes per hidden layer. I was unable to improve the model by including dropout layers, but I feel it can be improved and made more robust by adding dropout layers possibly in combination with adding one or more hidden layers. It’s also possible that instead of 2 hidden layers of 16 nodes, perhaps making the second hidden layer 8 nodes could produce better results. All of these options can only be explored through trial and error or using a combination of a DNN and another learning algorithm to optimize its architecture and hyper-parameters. Some of these optimization techniques have already been discussed in other research papers [5][6].

It would also be interesting if other patient features could be added to the feature set. It is possible that there are other bio indicators that can be added to the data set to improve the classification models.

## References

[1]: <https://www.breastcancer.org/symptoms/understand_bc/statistics>.

[2]: W.N. Street, W.H. Wolberg and O.L\*. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.

[3]: W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) 163-171.

[4]: W.H. Wolberg, W.N. Street, D.M. Heisey, and O.L. Mangasarian. Computerized breast cancer diagnosis and prognosis from fine needle aspirates. Archives of Surgery 1995;130:511-516.

[5]: Samanta, B.\*, Banopadhyay, S.\*\*, Ganguli, R.\*\* & Dutta, S. A comparative study of the performance of single neural network vs. Adaboost algorithm based combination of multiple neural networks for mineral resource estimation. Journal of the Southern African Institute of Mining and Metallurgy, Volume 105, Number 4, 1 April 2005, pp. 237-246(10)

[6]: T. Windeatt, R.Ghaderi. AdaBoost and neural networks. ESANN'1999 proceedings - European Symposium on Artificial Neural Networks. Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 123-128