1 Introduction

The data contains descriptive attributes of digitized images of a process known as fine needle aspirate (**FNA**) of breast mass. We have a total of 29 features that were computed for each cell nucleus with an ID Number and the Diagnosis (**malignant** and **benign**).

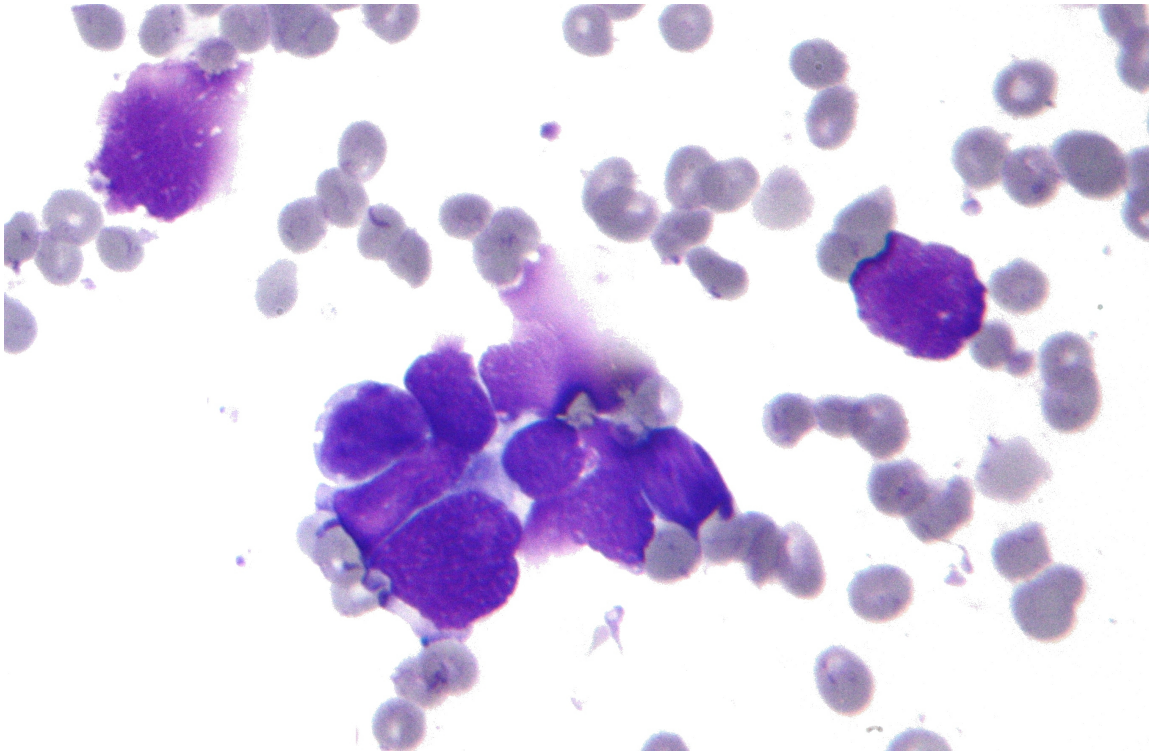


Image of small cell carcinoma

Source: By Nephron [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0) or GFDL (http://www.gnu.org/copyleft/fdl.html)], from Wikimedia Commons (https://commons.wikimedia.org/wiki/File:Small_cell_lung_cancer_-_cytology.jpg)

1.1 Project Extension: BIOLOGICAL MASS DATA

The following tutorial will cover the main ideas for analyzing a typical data-set. However, after this has been done, the next step would be to incorporate OMICS MASS DATA. For example, you could be given genomics sequencing data for each patient from which you can infer even more information. The main learning effect you should get from that is the following:

In bioinformatics (or computational biology) you very often need to analyze clinical data-sets, mainly to classify a patient's status or to derive disease-specific features. What is crucial to understand is that this data might look "small", but many of the variables are actually coming from some "big" data analysis, where potentially many TBs of raw data has been analyzed. This could be some omics analysis, or some image analysis, where one image often has several GBs.

PART ONE: Exploratory Analysis

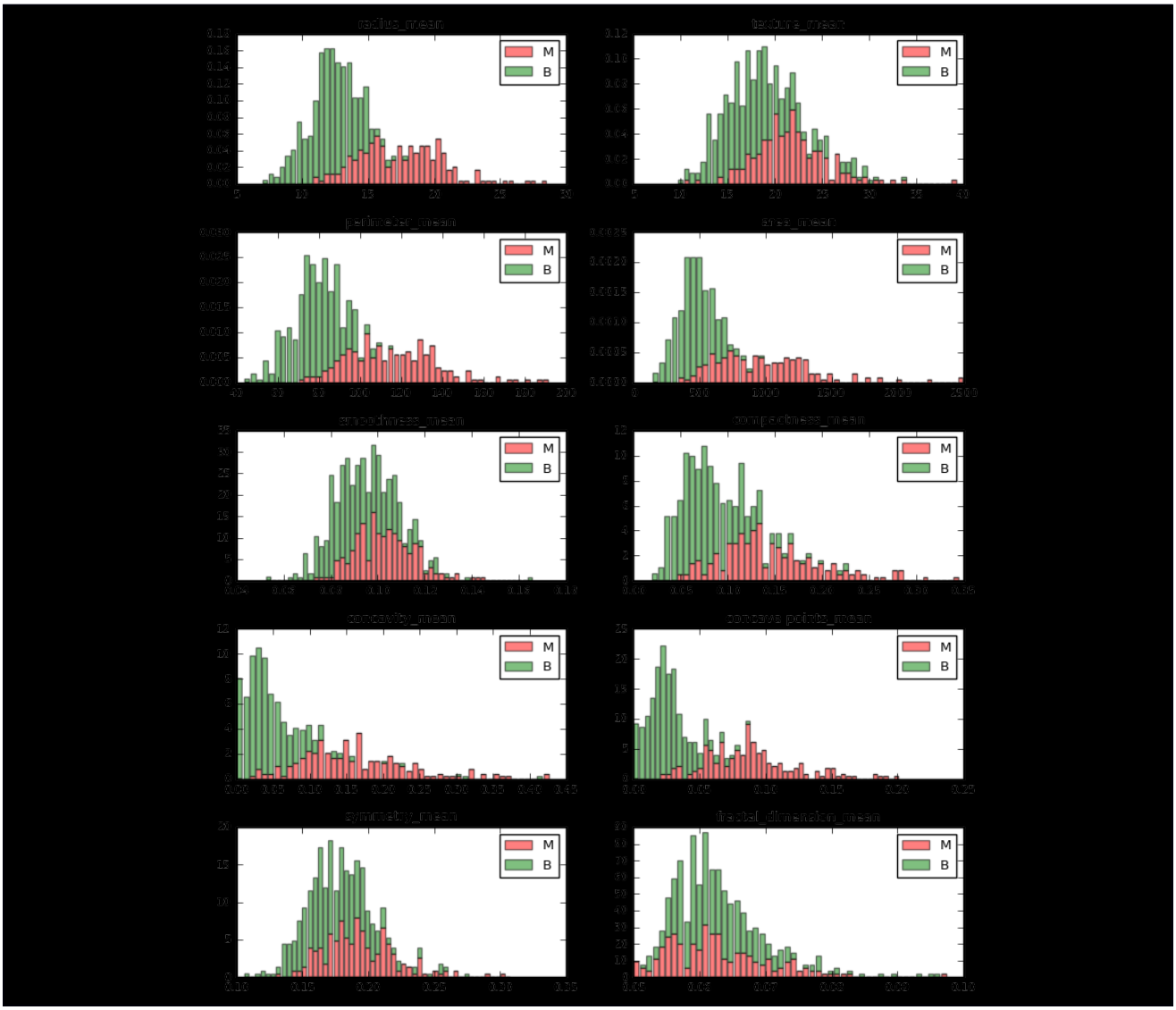
2 PART I: Exploratory Analysis

An important process in **Machine Learning** is doing an **Exploratory Analysis** to get a *feel* for your data. Creating visuals can help people understand the data-set and allow for digestable pieces of information that sometimes code and predictive analytics wouldn't allow. Often times we will try to jump into the predictive modeling, but it helps us create narratives which will allow us to give context to people who are not as driven by data.

3 Visual Exploratory Analysis

It is always a good idea to plot the individual fratures of the data with respect to the available classes to get a first expression about the distributions. The following plot gives a first impression whether we could tell apart the two groups (shown in green and red) just based on individual features.

3.1 Overview: Nucleus Features vs. Diagnosis



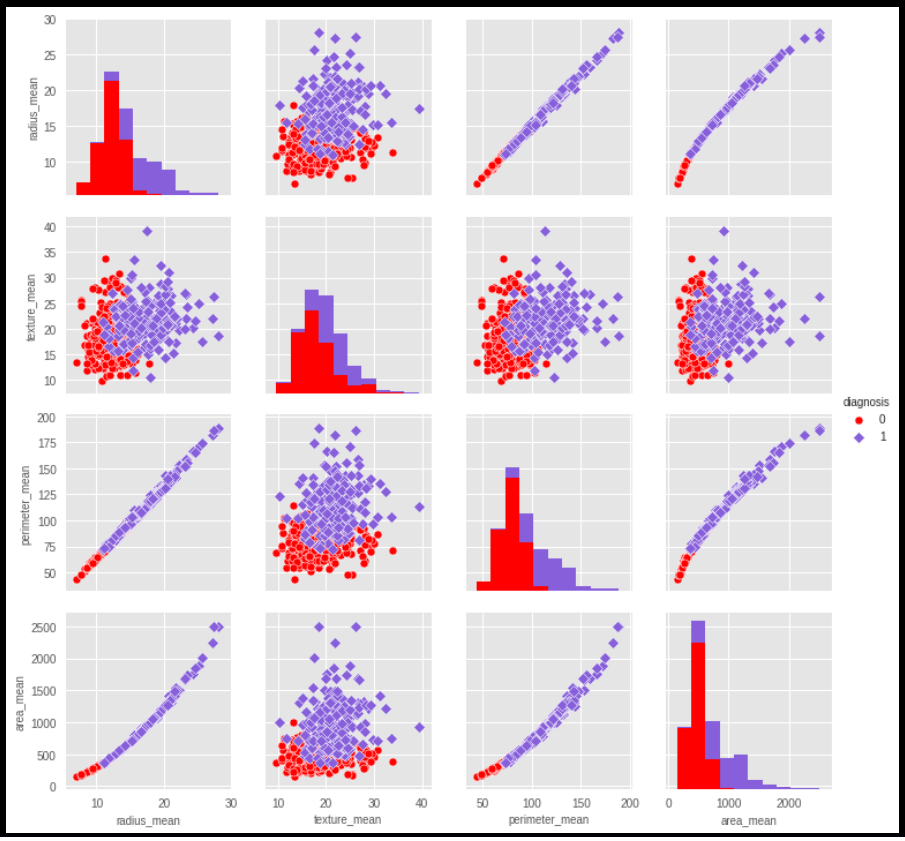
3.1.1 *Observations*

1. The mean values of cell radius, perimeter, area, compactness, concavity and concave points can be used in classification of the cancer. Larger values of these parameters tends to show a correlation with malignant tumors.
2. The mean values of texture, smoothness, symmetry or fractal dimension do not show a particular preference of one diagnosis over the other. In any of the histograms there are no noticeable large outliers that warrants further cleanup.

3.2 Scatterplot Matrix

For this visual we used some variables that were indicators of being influential from the last analysis.

Terminal Output



You see a matrix of the visual representation of the relationship between the first 4 “mean” variables.

Within each scatterplot we can color the two classes of **Diagnosis**, which we can clearly see that we can sometimes easily distinguish the difference between **Malignant** and **Begnin**. As well as some variable interactions have an almost linear relationship.

Of course these are just 2-dimensional representations, but it is still interesting to see how variables interact with each other in our data-set.

3.3 Pearson Correlation Matrix

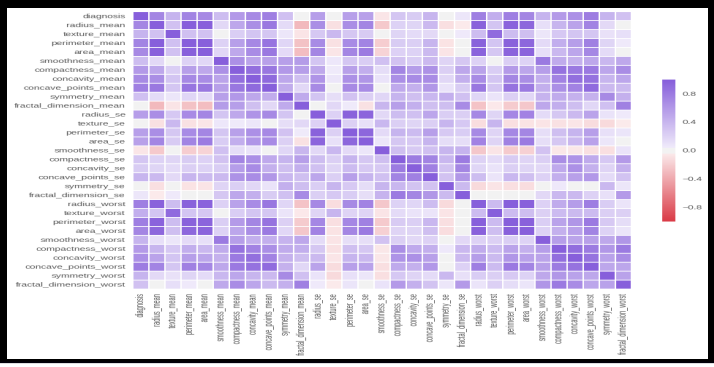
The next visual gives similar context that the last visual provided, and it is called the *Pearson Correlation Matrix*.

Variable correlation within a *Machine Learning* context doesn't play as an important role as say *linear regression*, but there can still be some dangers when a data-set has too many correlated variables.

When two features (or more) are almost perfectly correlated in a *Machine Learning* setting then the inclusion of these features does not add addition information to your process. This then has the potential to hurt your algorithm's accuracy, since we are potentially utilizing a large feature space that can cause what is known as the Curse of Dimensionality. Thus feature extraction would help reduce the amount of *noise* in your feature space, see *principal component analysis*, or *t-distributed stochastic neighbor embedding*.

Many of our algorithms are also very computationally expensive, so utilizing a dimension reduction algorithm would also help performance and computational time.

Terminal Output

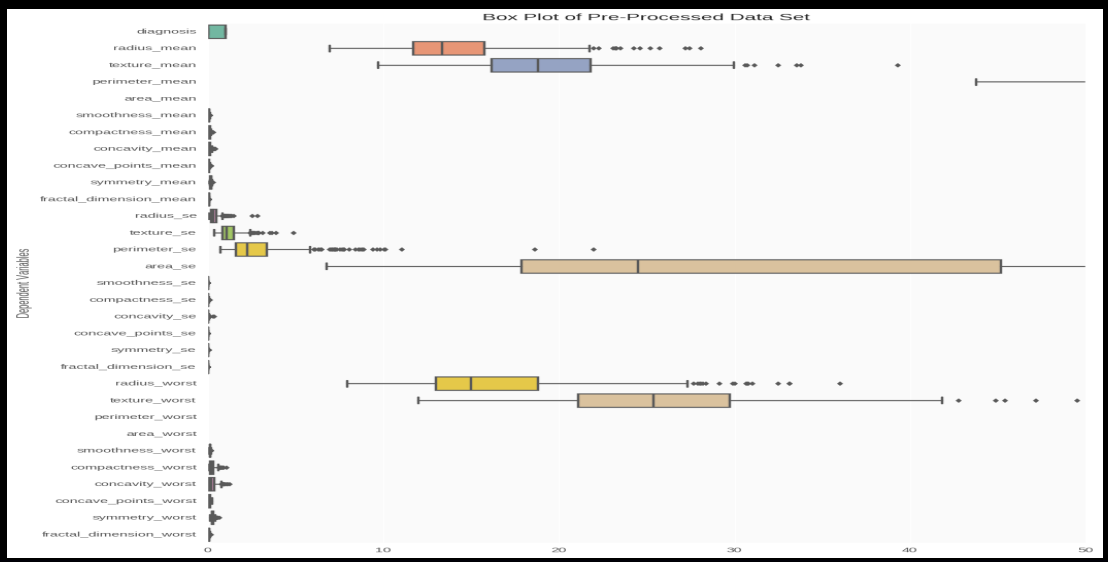


We can see that our data-set contains mostly positive correlation, as well as re-iterating to us that the 5 dependent variables we featured in the *Scatterplot Matrix* have strong *correlation*. Our variables don't have too much correlation so we won't go about doing feature extraction processes like *Principal Component Analysis* (**PCA**), but you are more welcomed to do so (you will probably get better prediction estimates).

3.4 Boxplots

Next, we will look at boxplots of the data to show the high variance in the distribution of our variables. This will help drive home the point of the need to do some appropriate transformation for some models we will be employing. This is especially true for *Neural Networks*.

Terminal Output



Not the best picture but this is a good starting point for the next step in our *Machine learning* process.

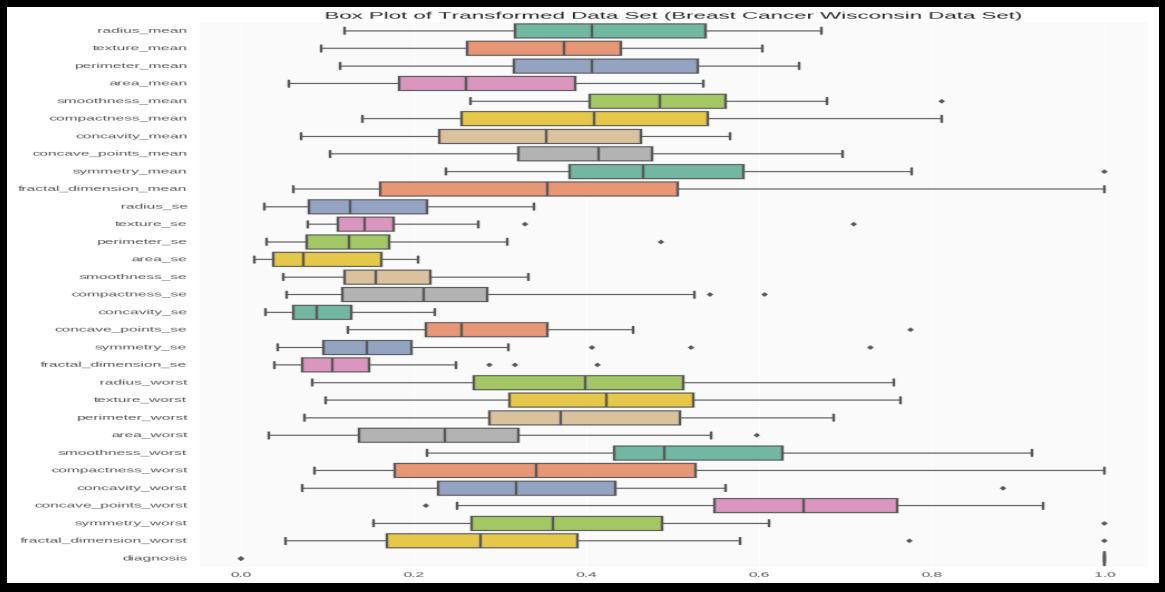
3.4.1 Normalization

Here we used a custom function to set the minimum of 0 and maximum of 1 to help with some machine learning applications later on in this report. Notice that we will use this function only for the visualization of my data-set. Important to note because if we were to use this transformation, during my machine learning process we would be guilty of the process called, *data leakage*, more on this later in the *neural networks* section.

3.4.2 Box Plot of Transformed Data

Now to further illustrate the transformation let's create a *boxplot* of the scaled data-set, and see the difference from our first *boxplot*.

Terminal Output



There are different forms of transformations that are available for *machine learning* and we suggest you research them to gain a better understanding as to when to use a transformation. But for this project we will only employ the transformed dataframe on *Neural Networks*.

Part II: Machine Learning

4 PART II: Machine Learning for Predictive Modeling

In the following, we will use a well-established **Machine Learning** techniques, knows as (Bagging) Random Forrest build a model that learn from our data and generate a model that allows prediction of unknown data.

4.1 Prerequisites: Training and Test Sets

Now that we've gotten a *feel* for the data in the previous part, we are ready to begin predictive modeling. When going about predictive modeling, especially when new data isn't readily available, creating a *training* and *test sets* will help in understanding your model's predictive power.

We will use the *training set* to train our model, essentially learning from data it's seeing to later infer data it hasn't seen yet. We want to avoid using our entire data-set on the training process, due to the process called *overfitting*.

We run the risk of *over-fitting* our data when we train the model on our entire data-set, especially true for this data-set since we don't have any other data to see how well our data does. *Over-fitting* will cause our model to output strong predictive power, but only for our training data usually performing poorly on new data.

We avoid this through the use of the *training* and *test set*, where we measure the predictive power on the *test set*. This will be a good indicator of our model's performance, but test sets also have their limitations. This is where *Cross Validation* will come into play, but more on this later.

We split the data-set into our training and test sets which will be (pseudo) randomly selected having a *80-20%* split.

NOTE: What we mean when we say pseudo-random is that we would want everyone who replicates this project to get the same results especially if you're trying to learn from this project. So we use a random seed generator and set it equal to a number of our choosing, this will then make the results the same for anyone who uses this generator, awesome for reproducibility.

5 Random Forest

Also known as *Random Decision Forest*, *Random Forest* is an entire forest of random uncorrelated decision trees. This is an extension of *Decision Trees* that will perform significantly better than a single tree because it corrects over-fitting. Here is a brief overview of the evolution of *CART* analysis:

- Single Decision Tree (Single tree)
- Bagging Trees (Multiple trees) [Model with all features, M, considered at splits, where M = all features]
- Random Forest (Multiple trees) [Model with m features considered at splits, where m < M, typically m = sqrt(M)]

5.1 Bagging Trees

Decision Trees tend to have *low bias and high variance*, a process known as *Bagging Trees* (*Bagging = Bootstrap Aggregating*) was an extension that does random sampling with replacement where after creating N trees it classifies on majority votes. This process reduces the variance while at the same time keeping the bias low. However, a downside to this process is if certain features are strong predictors then too many trees will employ these features causing correlation between the trees.

Thus *Random Forest* aims to reduce this correlation by choosing only a subsample of the feature space at each split. Essentially aiming to make the trees more independent thereby reducing the variance.

Generally, we aim to create 500 trees and use our m to be sqrt(M) rounded down. So since we have 30 features we will use 5 for my `max_features` parameter. we will be using the *Entropy Importance* metric.

For this model we use an index known as the Information Gain.

5.2 Variable Importance

A very useful feature from the tree analysis is that it allows extracting which features where the most important one. This is usually done using the concepts of *Entropy* and *information gain*. Thus, we can derive for each variable their (relative) *variable importance* and gives us and idea about which variables played an important role in the specific (!) forest that was created in the previous step.

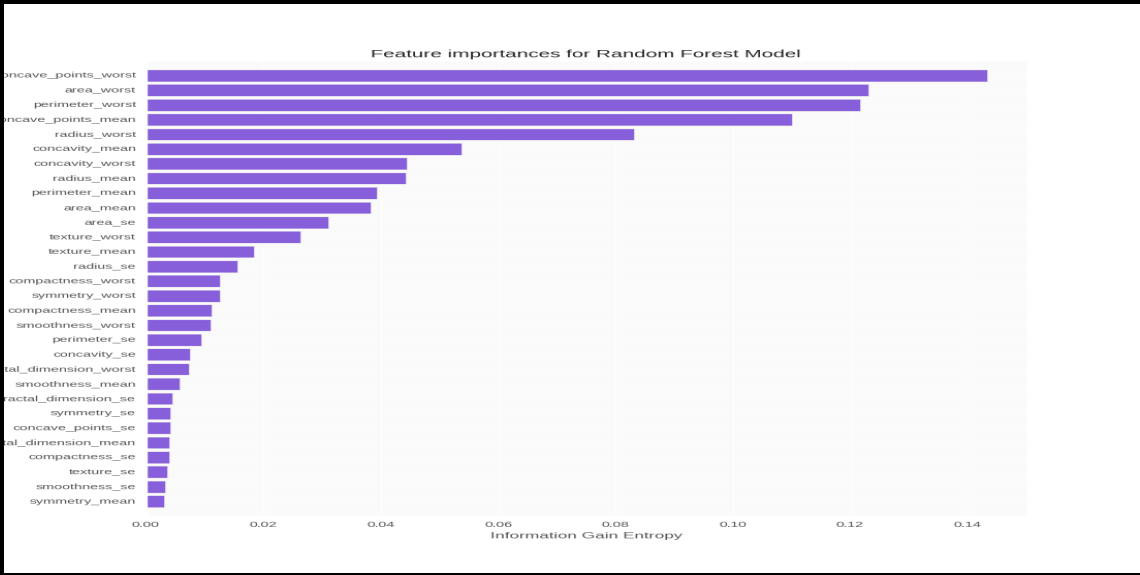
Let's start by creating the forest and then ordering all our variables (features), by importance using the *entropy* criterion.

Terminal Output

```
Feature ranking:
1. The feature 'concave_points_worst' has a Information Gain of 0.143349
2. The feature 'area_worst' has a Information Gain of 0.123030
3. The feature 'perimeter_worst' has a Information Gain of 0.121727
…
29. The feature 'smoothness_se' has a Information Gain of 0.003234
30. The feature 'symmetry_mean' has a Information Gain of 0.003022
```

5.2.1 Feature Importance Visual

Terminal Output



5.3 Cross Validation

We employ a 10 fold *cross validation* method to get an accuracy estimation for our model.

Terminal Output

```
Accuracy:  0.963 (+/-  0.013)
```

5.4 Conclusions for Random Forest

Our *Random Forest* performed pretty well, we have a personal preference to tree type models because they are the most interpretable and give insight to data that some other models don't (for instance **K-NN**). We were able to see which variables were important when the *random forest* was created and thus we can expand on our data/model if we choose too, through the variable importance of *random forest*. That can be an exercise for later on to see if choosing a subset of the data-set will help in prediction power. For now we am content with using the entire data-set for instructive purposes.

Important to note, is that the *random forest* performed better in terms of *false negatives*, only slightly though.

6 Conclusions of Part II

When choosing models, it isn't just about having the best accuracy, we are also concerned with insight because these models can help tell us which dependent variables are indicators thus helping researchers in the respective field to focus on the variables that have the most statistically significant influence in predictive modeling within the respective domain. Although this data-set was engineered to facilitate machine learning processing employing the methodologies, we would say using both **Random Forest** and **Neural Networks** would help in predicting and getting insight into the data.