Classification of Cancer Cells Using Machine Learning Model (SVM)

Arunima Varma

September 22, 2020

1. Introduction

1.1 Background

Breast cancer is the most common cancer amongst women in the world. It accounts for 25% of all cancer cases, and affected over 2.1 Million people in 2015 alone. These cells usually form tumours that can be seen via X-ray or felt as lumps in the breast area.

Early diagnosis significantly increases the chances of survival. The key challenges upon detection is how to classify tumours into malignant or benign. A tumour is considered malignant if the cells can grow into surrounding tissues or spread to distant areas of the body. A benign tumour does not invade nearby tissue nor spread to other parts of the body the way cancerous tumours can.

Machine Learning is a sub-field of Artificial Intelligence that gives systems the ability to learn themselves without being explicitly programmed to do so. Machine Learning can be used in solving many real-world problems. ML techniques can dramatically improve the level of diagnosis in breast cancer. Research shows that experienced physicians can detect cancer by 79% accuracy, while a 91% accuracy can be achieved using ML techniques.

1.2 Problem

In this study, my task is to classify tumours into malignant (cancerous) or benign (non-cancerous) using features obtained from several cell images.

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

2. Data Acquisition and Cleaning

2.1 Data Sources

Scikit-learn comes with a few small standard datasets that do not require downloading any file from any external website. The dataset that I will be using for our machine learning problem is the Breast cancer Wisconsin (diagnostic) dataset. The dataset includes several data about the breast cancer tumours along with the classification labels, viz., malignant or benign.

2.2 Data Cleaning

Attribute Information:

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

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

1. Radius (mean of distances from centre to points on the perimeter)
2. Texture (standard deviation of grey-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)

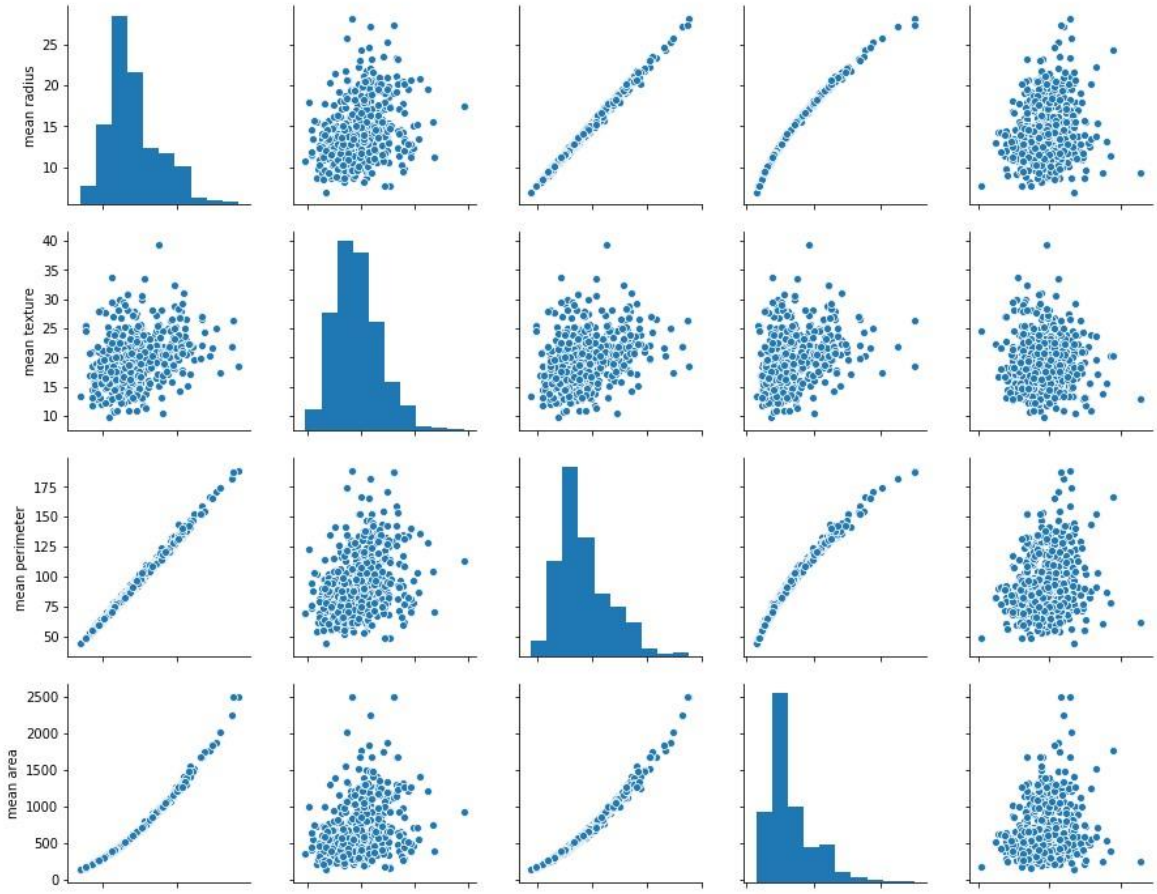
2.3 Feature Selection

The important attributes that we must consider from that dataset are ‘target-names'(the meaning of the labels), ‘target'(the classification labels), ‘feature names'(the meaning of the features) and ‘data'(the data to learn).

To get a better understanding of what the dataset contains and how we can use the data to train our model, let us first organize the data and then see what it contains.

So, we see that each dataset of a tumour is labelled as either ‘malignant’ or ‘benign’. From here, each label is linked to binary values of 0 and 1, where 0 represents malignant tumors and 1 represents benign tumours.
Here, there are 30 features or attributes that each dataset of the tumor has. We will be using the numerical values of these features in training our model and make the correct prediction, whether or not a tumor is malignant or benign, based on these features.

3. Visualize the relationship between our features :

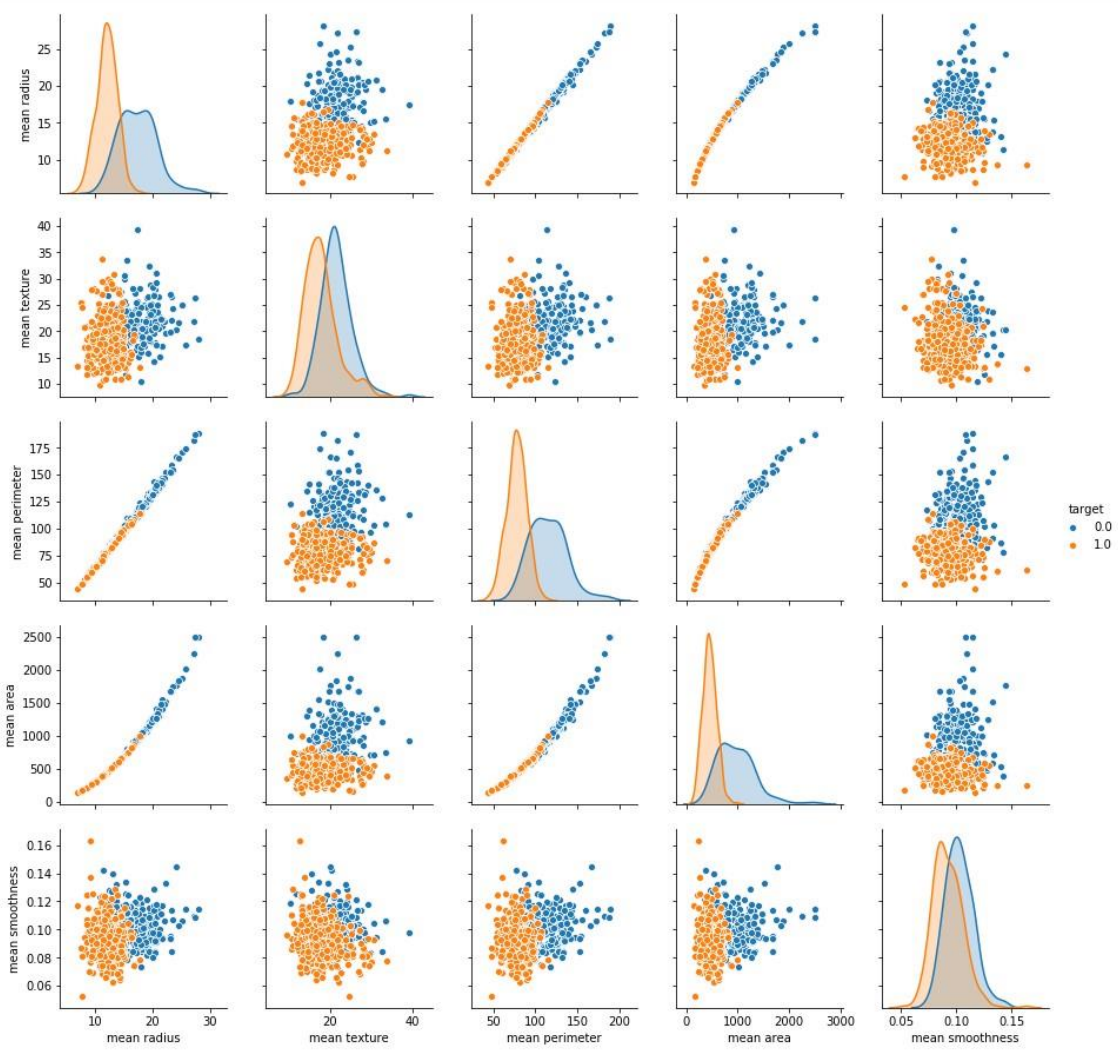


The relationship between the features: ‘mean radius’, 'mean texture', 'mean perimeter', 'mean area', 'mean smoothness' is plotted as shown.

Note:

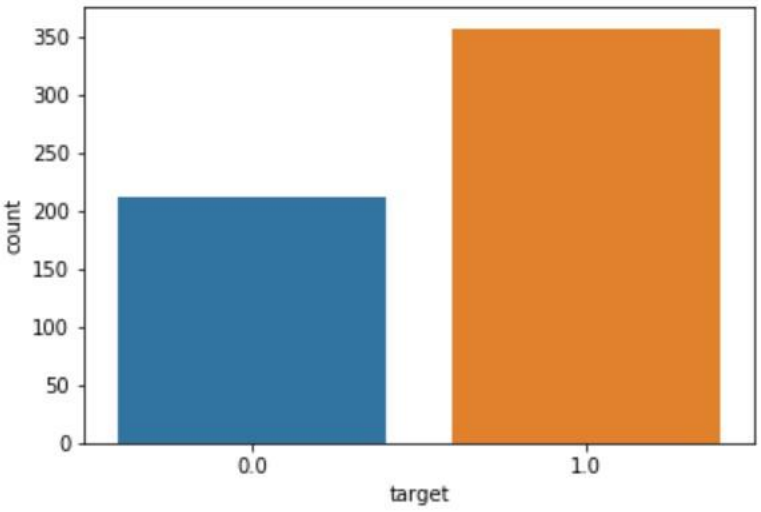
1.0 (Orange) = Benign (No Cancer)

0.0 (Blue) = Malignant (Cancer)



How many Benign and Malignant do we have in our dataset?
This can be visualized as follows:

```
: sns.countplot(df_cancer['target'], label = "Count")
47]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6d439da390>
```



Note:

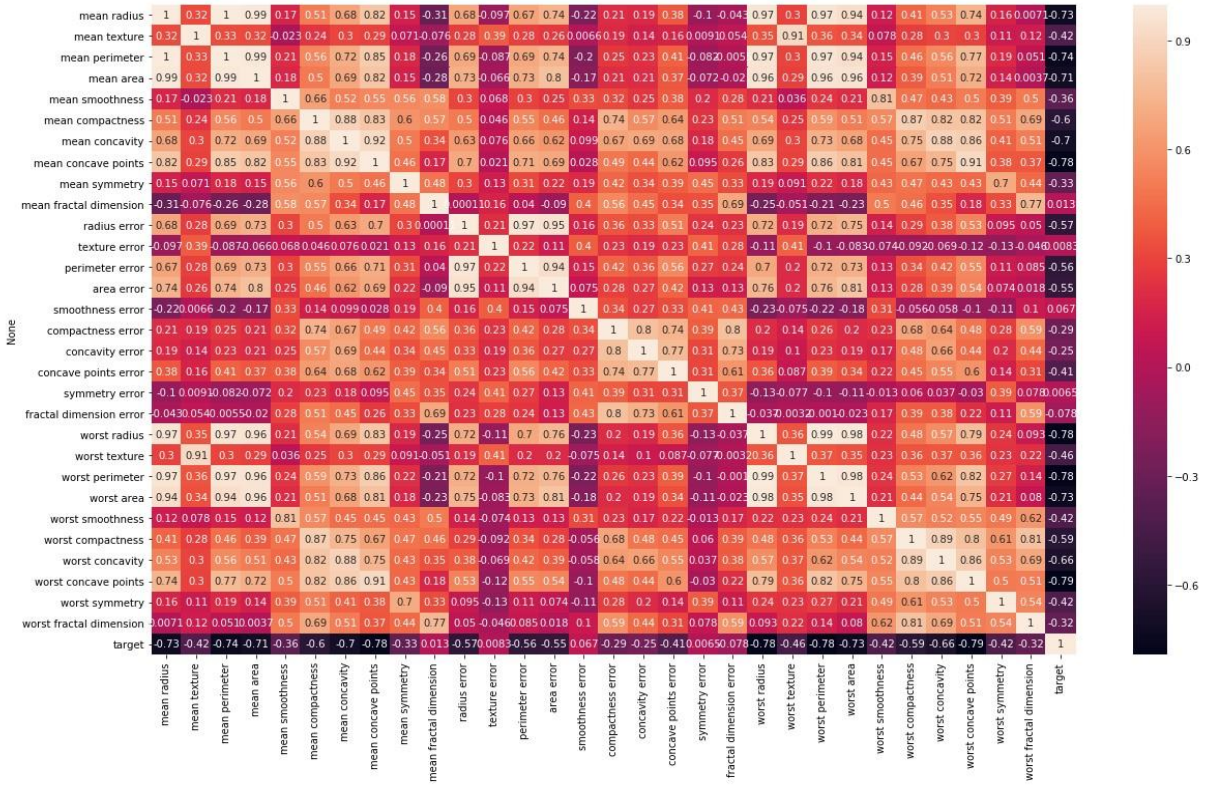
1.0 (Orange) = Benign (No Cancer)

0.0 (Blue) = Malignant (Cancer)

Let's check the correlation between our features

```
plt.figure(figsize=(20,12))
sns.heatmap(df.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6e3ecab320>
```



Correlation is a statistical term which in common usage refers to how close two variables are to having a linear relationship with each other. Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have high correlation, we can drop one of the two features.

4. Predictive Modeling

As data scientists we attempt to make our understanding of relationships between different quantities more precise through using data and mathematical/statistical structures.

This process is called **modelling.**

Models are simplifications of reality that help us to better understand that which we observe.

In a data science setting, models generally consist of an independent variable (or output) of interest and one or more dependent variables (or inputs) believed to influence the independent variable.

- **Inference** is judging what the relationship, if any, there is between the data and the output.
- **Prediction** is making guesses about future scenarios based on data and a model constructed on that data.

4.1 Introduction to Classification Modeling: Support Vector Machine (SVM)

SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handle nonlinear input spaces. It is used in a variety of applications such as face detection, intrusion detection, classification of emails, news articles and web pages, classification of genes, and handwriting recognition.

SVM is an exciting algorithm and the concepts are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points.

Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane (MMH) that best divides the dataset into classes.

4.2 How does SVM work?

The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps:

- Generate hyperplanes which segregates the classes in the best way. Left-hand side figure showing three hyperplanes black, blue and orange. Here, the blue and orange have higher classification error, but the black is separating the two classes correctly.
- Select the right hyperplane with the maximum segregation from the either nearest data points.

4.3 SVM Kernels

The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space. Here I have used Linear Kernelling.

Linear Kernel A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

4.4 Loading Data

Let's first load the required dataset you will use.

```
df_cancer.head()
```

Out[40]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	conc
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.

5 rows × 31 columns

4.5 Exploring Data

After we have loaded the dataset, we want to know a little bit more about it. We can check feature and target names. Let's take a look at the target set.

```
# print the cancer labels (0:malignant, 1:benign)
print(cancer.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

4.6 Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Split the dataset by using the function train_test_split(). We need to pass 3 parameters features, target, and test_set size. Additionally, we can use random_state to select records randomly.

4.7 Generating Model

To build support vector machine model. First, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in SVC() function.

Then, fit the model on train set using fit() and perform prediction on the test set using predict().

Then, fit the model on train set using fit() and perform prediction on the test set using predict().

17]:
```
#Import svm model
from sklearn import svm

#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

5. Evaluating the Model

Let's estimate how accurately the classifier or model can predict the breast cancer in patients. Accuracy can be computed by comparing actual test set values and predicted values.

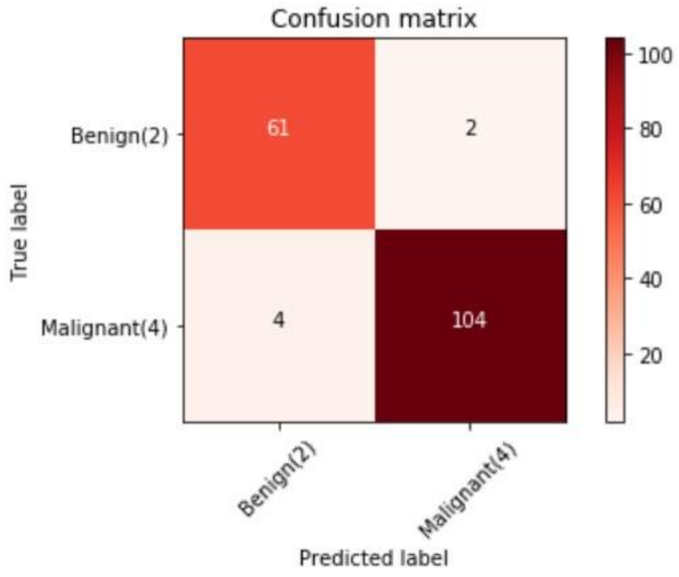
Next step is to check the accuracy of our prediction by comparing it to the output we already have (y_pred). We are going to use confusion matrix for this comparison. Well, we have got a classification rate of 96.49%, considered as very good accuracy.

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.95        63
           1       0.98      0.96      0.97       108

   micro avg       0.96      0.96      0.96       171
   macro avg       0.96      0.97      0.96       171
weighted avg       0.97      0.96      0.97       171

Confusion matrix, without normalization
[[ 61   2]
 [  4 104]]
```



We can also use the **f1_score** from sklearn library:

```
om sklearn.metrics import f1_score
int("f1_score is :",f1_score(y_test, y_pred, average='weighted'))
```

```
f1_score is : 0.9650224422036399
```

We'll try **jaccard index** for accuracy:

```
om sklearn.metrics import jaccard_similarity_score
int("jaccard index is: ",jaccard_similarity_score(y_test, y_pred))
```

```
jaccard index is:  0.9649122807017544
```

The Jaccard similarity index (sometimes called the Jaccard similarity *coefficient*) compares members for two sets to see which members are shared and which are distinct. It's a measure of similarity for the two sets of data, with a range from 0% to 100%. The higher the percentage, the more similar the two populations.

F1 Score (aka F-Score or F-Measure) – A helpful metric for comparing two classifiers. F1 Score takes into account precision and the recall. It is created by finding the the harmonic mean of precision and recall.

F1 = 2 x (precision x recall)/(precision + recall)

We have got a F1_score and Jaccard index rate of 96.49%, considered as very good accuracy.

6.Conclusion

6.1 The advantages of SVM

Effective in high dimensional spaces. Still effective in cases where number of dimensions is greater than the number of samples. Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

6.2 The disadvantages of SVM

If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial. SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation (see Scores and probabilities).

6.3 Summary

This project took us through the journey of explaining what “modelling” means in Data Science, difference between model prediction and inference, introduction to Support Vector Machine (SVM), advantages and disadvantages of SVM, training an SVM model to make accurate breast cancer classifications, improving the performance of an SVM model, and testing model accuracy using Confusion Matrix.

7. Future directions

Model in this study mainly focused on individual features. However, interactions with doctors, diagnostics team might also contribute to a classification or prediction. These data are obviously more difficult to extract and quantify, but if optimized, could bring significant improvements to the models.

Sources:

http://scikit-learn.org/stable/modules/svm.html

http://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf

http://pyml.sourceforge.net/doc/howto.pdf

https://www.bcrf.org/breast-cancer-statistics

https://www.cancer.org/cancer/breast-cancer/about/what-is-breast-cancer.html