VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Breast Cancer Classification Using Support Vector Machine (SVM)

Project Report

SUBMITTED TO

Dr.KANIMOZHI.G

BY

1.NITHIN S (19BEE1046)

2.SHYAM SUNDAR (19BEE1060)

3. SRI DHARSAN A (19BEE1069)

4.RAJESH R (19BEE1171)

SCHOOL OF ELECTRICAL ENGINEERING
VELLORE INSTITUTE OF TECHNOLOGY
CHENNAI,TAMILNADU-600127

Nov, 2021

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. In 2020, there were **2.3 million women diagnosed** with breast cancer and 685 000 deaths globally. As of the end of 2020, there were 7.8 million women alive who were diagnosed with breast cancer in the past 5 years, making it the world's most prevalent cancer. It starts when cells in the breast begin to grow out of control. These cells usually form tumors 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 against it's detection is how to classify tumors into malignant (cancerous) or benign(non cancerous). A tumor is considered malignant if the cells can grow into surrounding tissues or spread to distant areas of the body. A benign tumor does not invade nearby tissue nor spread to other parts of the body the way cancerous tumors can. But benign tumors can be serious if they press on vital structures such as blood vessels or nerves.

Breast Cancer, similar to other cancers, starts with a rapid and uncontrolled outgrowth and multiplication of a part of the breast tissue, which depending on its potential harm, is divided into benign and malignant types

Most breast cancers are found in women who are **50 years old or older**. Some women will get breast cancer even without any other risk factors that they know of. Having a risk factor does not mean you will get the disease, and not all risk factors have the same effect.

The most common cause of death was **metastatic disease to various organs**, accounting for 42% of all deaths. Infection was the second most common cause of death; however, only 27% of the patients with infection had significant neutropenia. In patients dying of hemorrhage, only 9% were thrombocytopenic.

INTRODUCTION:

Machine Learning technique can dramatically improve the level of diagnosis in breast cancer. Research shows that experienced physicians can detect cancer by 79% accuracy, while a 91 %(sometimes up to 97%) accuracy can be achieved using Machine Learning techniques.

In this project, our task is to classify tumors 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.

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 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)

Introduction to Classification Modeling: Support Vector Machine (SVM)

What is a Support Vector Machine (SVM)?

A Support Vector Machine (SVM) is a binary linear classification whose decision boundary is explicitly constructed to minimize generalization error. It is a very powerful and versatile Machine Learning model, capable of

performing linear or nonlinear classification, regression and even outlier detection.
SVM is well suited for classification of complex but small or medium sized datasets.

How does SVM classify?

It's important to start with the intuition for SVM with the **special linearly separable** classification case.
If classification of observations is **"linearly separable"**, SVM fits the **"decision boundary"** that is defined by the largest margin between the closest points for each class. This is commonly called the **"maximum margin hyperplane (MMH)"**.

How does SVM classify?

It's important to start with the intuition for SVM with the **special linearly separable** classification case.

If classification of observations is **"linearly separable"**, SVM fits the **"decision boundary"** that is defined by the largest margin between the closest points for each class. This is commonly called the **"maximum margin hyperplane (MMH)"**.

The advantages of support vector machines are:

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.

Coding Part:

Loading Python Libraries and Breast Cancer Dataset:

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

        %matplotlib inline

        #Import Cancer data from the Sklearn library
        # Dataset can also be found here (http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29)

        from sklearn.datasets import load_breast_cancer
        cancer = load_breast_cancer()
```

Let’s view the data in a dataframe:

```
In [3]: df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']], columns = np.append(cancer['feature_names'], ['target

        df_cancer.head()
```

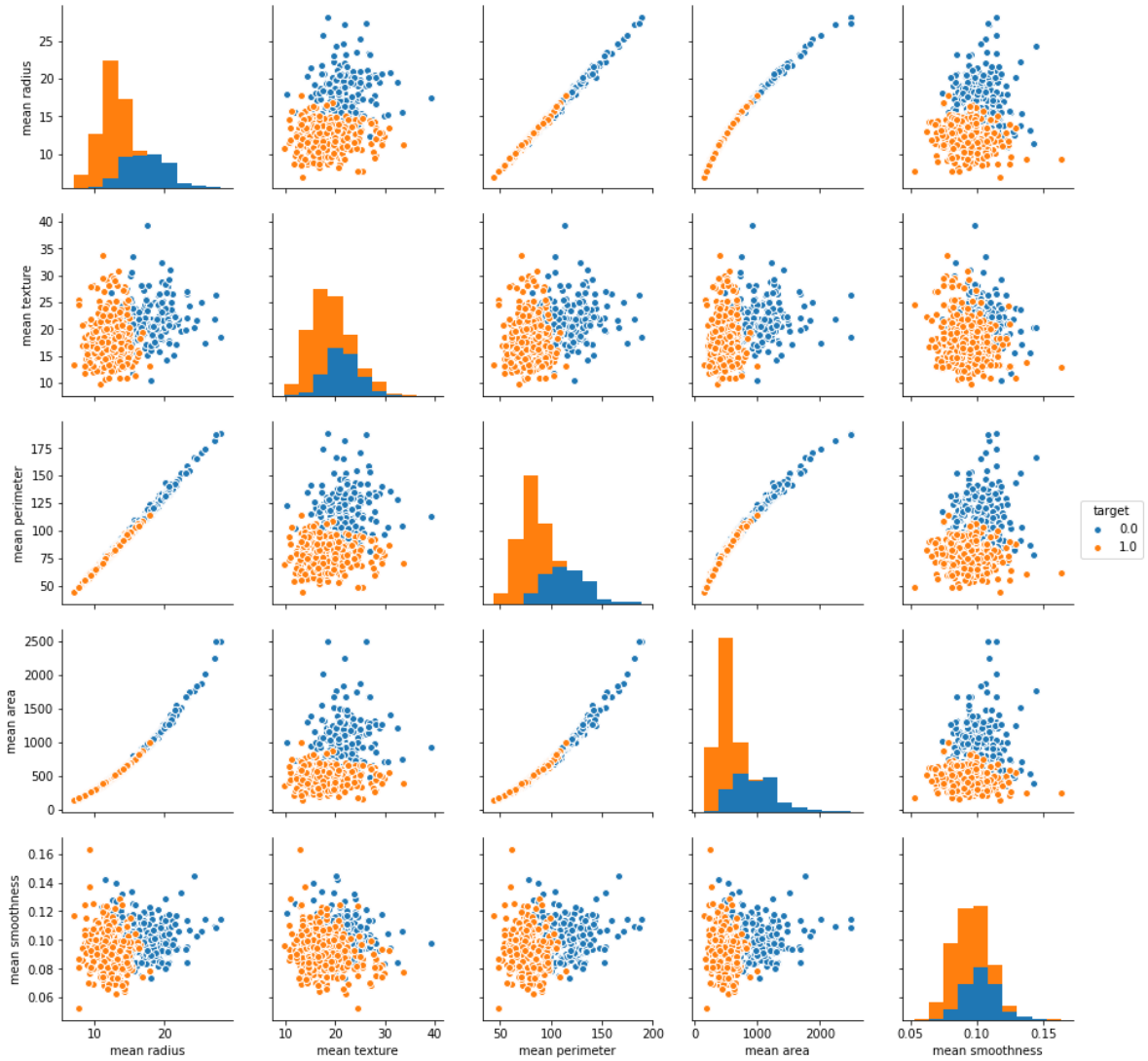
Out[3]:

	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	comp
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	
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	
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	
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	
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	

5 rows × 31 columns

Visualize the relationship between our features:

```
In [7]: # Let's plot out just the first 5 variables (features)
        sns.pairplot(df_cancer, hue = 'target', vars = ['mean radius', 'mean texture', 'mean perimeter','mean area','mean smoothl
```



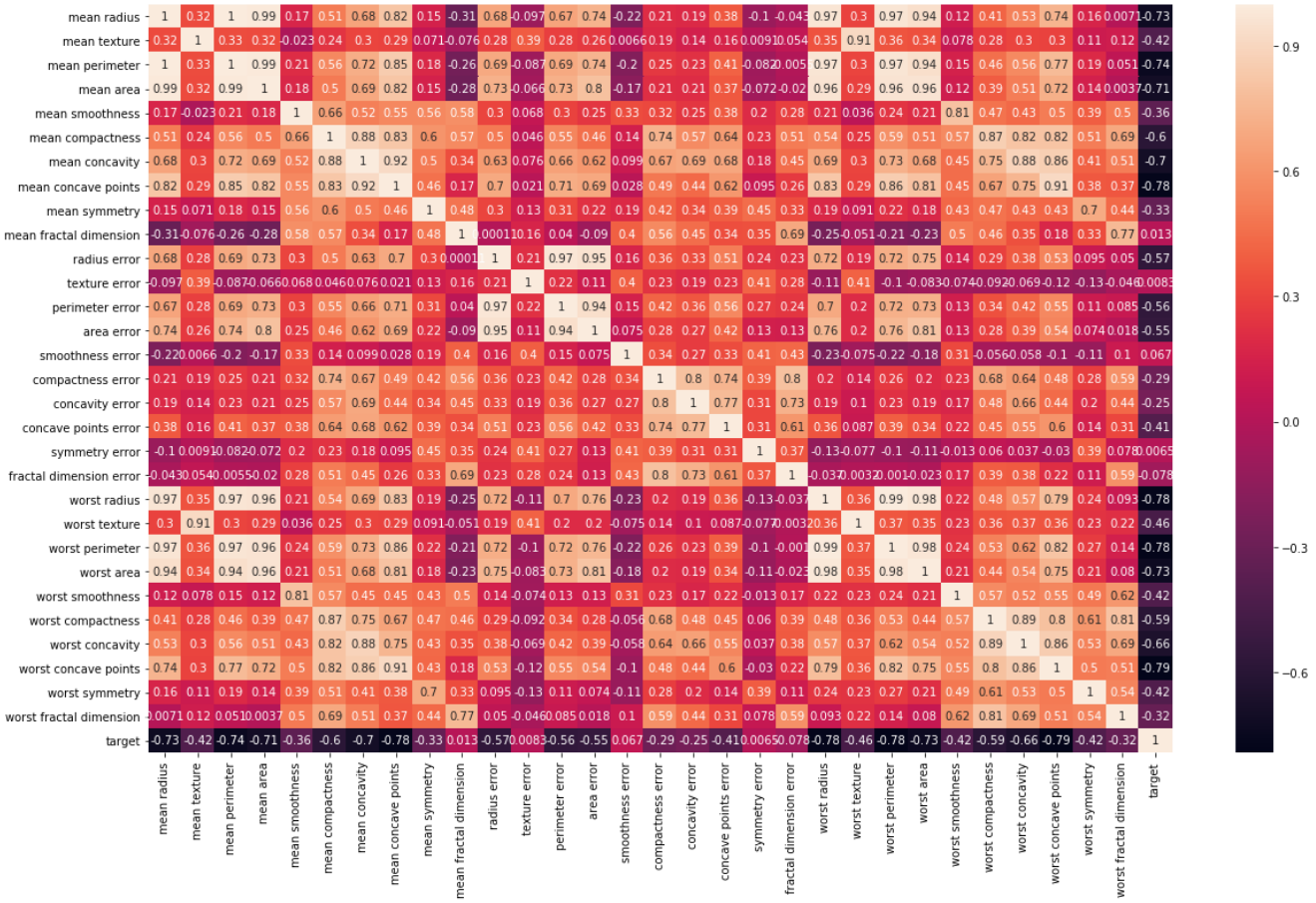
Note:

1.0 (Orange) = Benign (No Cancer)

0.0 (Blue) = Malignant (Cancer)

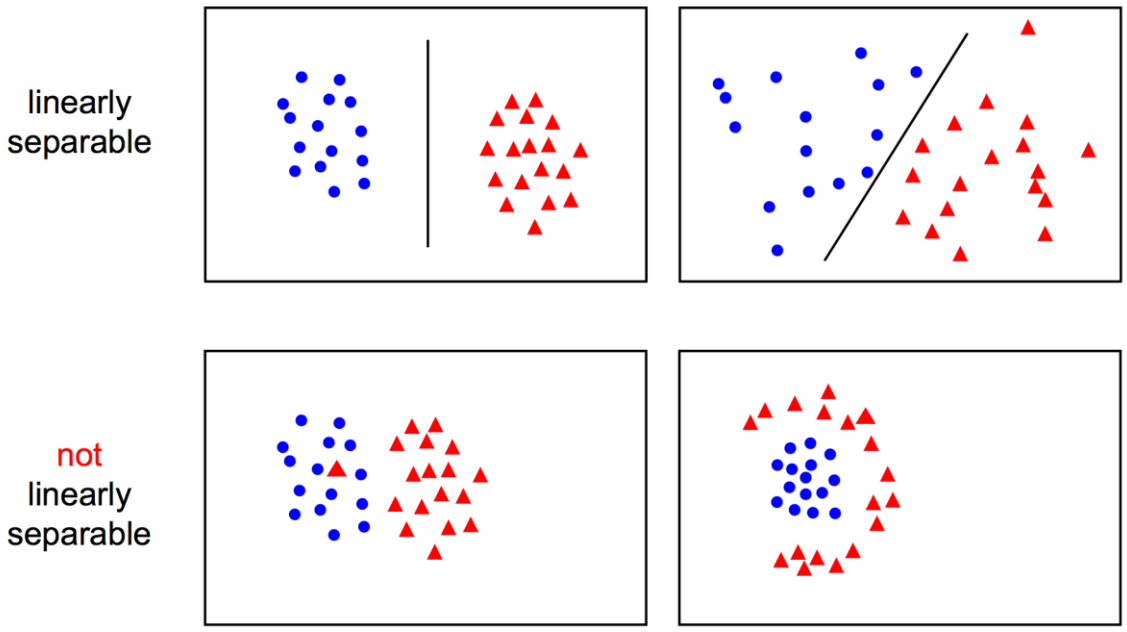
Let’s check the correlation between our features:

```
In [10]: plt.figure(figsize=(20,12))
         sns.heatmap(df_cancer.corr(), annot=True)
```



There is a strong correlation between mean radius and mean perimeter, as well as mean area and mean perimeter

SVM Sample classification



Model Training

From our dataset, let's create the target and predictor matrix

"y" = Is the feature we are trying to predict (Output). In this case we are trying to predict if our "target" is cancerous (Malignant) or not (Benign). i.e. we are going to use the "target" feature here.

"X" = The predictors which are the remaining columns (mean radius, mean texture, mean perimeter, mean area, mean smoothness, etc.)

```
In [11]: X = df_cancer.drop(['target'], axis = 1) # We drop our "target" feature and use all the remaining features in our dataf
         X.head()
```

Out[11]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area	wors smoothnes
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2019.0	0.162
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1956.0	0.123
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1709.0	0.144
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	567.7	0.209
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1575.0	0.137

5 rows × 30 columns

```
In [12]: y = df_cancer['target']
         y.head()
```

```
Out[12]: 0    0.0
         1    0.0
         2    0.0
         3    0.0
         4    0.0
         Name: target, dtype: float64
```

Create the training and testing data

Now that we've assigned values to our "X" and "y", the next step is to import the python library that will help us split our dataset into training and testing data.

Training data = the subset of our data used to train our model.

Testing data = the subset of our data that the model hasn't seen before (We will be using this dataset to test the performance of our model).

```
In [13]: from sklearn.model_selection import train_test_split
```

Let's split our data using 80% for training and the remaining 20% for testing.

```
In [14]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 20)
```

Let now check the size our training and testing data.

```
In [15]: print ('The size of our training "X" (input features) is', X_train.shape)
         print ('\n')
         print ('The size of our testing "X" (input features) is', X_test.shape)
         print ('\n')
         print ('The size of our training "y" (output feature) is', y_train.shape)
         print ('\n')
         print ('The size of our testing "y" (output features) is', y_test.shape)
```

```
The size of our training "X" (input features) is (455, 30)


The size of our testing "X" (input features) is (114, 30)


The size of our training "y" (output feature) is (455,)


The size of our testing "y" (output features) is (114,)
```

Import Support Vector Machine (SVM) Model

```
In [16]: from sklearn.svm import SVC
```

```
In [17]: svc_model = SVC()
```

Now, let's train our SVM model with our "training" dataset.

```
In [18]: svc_model.fit(X_train, y_train)
```

```
Out[18]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)
```

Let's use our trained model to make a prediction using our testing data

```
In [19]: y_predict = svc_model.predict(X_test)
```

Next step is to check the accuracy of our prediction by comparing it to the output we already have (y_test). We are going to use confusion matrix for this comparison.

The confusion matrix

The confusion matrix is a table representing the performance of your model to classify labels correctly.

A confusion matrix for a binary classification task:

	Predicted Negative	Predicted Positive
Actual Negative	True Negative (TN)	False Positive (FP)
Actual Positive	False Negative (FN)	True Positive (TP)

In a binary classifier, the "true" class is typically labeled with 1 and the "false" class is labeled with 0.

> **True Positive**: A positive class observation (1) is correctly classified as positive by the model.
>
> **False Positive**: A negative class observation (0) is incorrectly classified as positive.
>
> **True Negative**: A negative class observation is correctly classified as negative.
>
> **False Negative**: A positive class observation is incorrectly classified as negative.

Columns of the confusion matrix sum to the predictions by class. Rows of the matrix sum to the actual values within each class. You may encounter confusion matrices where the actual is in columns and the predicted is in the rows: the meaning is the same but the table will be reoriented.

> **Note:** Remembering what the cells in the confusion matrix represents can be a little tricky. The first word (True or False) indicates whether or not the model was correct. The second word (Positive or Negative) indicates the *model's guess* (not the actual label!).

Let’s create a confusion matrix for our classifier’s performance on the test dataset.

```
In [20]: # Import metric libraries

         from sklearn.metrics import classification_report, confusion_matrix
```

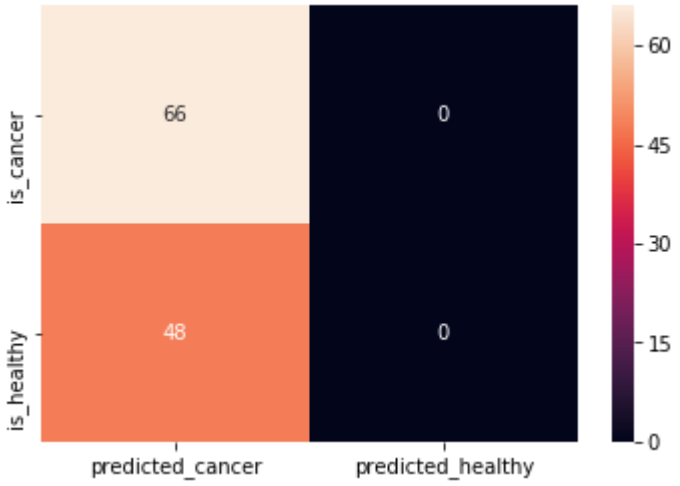
```
In [21]: cm = np.array(confusion_matrix(y_test, y_predict, labels=[1,0]))
         confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                                  columns=['predicted_cancer','predicted_healthy'])
         confusion
```

Out[21]:

	predicted_cancer	predicted_healthy
is_cancer	66	0
is_healthy	48	0

Let’s visualize our confusion matrix on a Heatmap

```
In [22]: sns.heatmap(confusion, annot=True)
```




```
In [23]: print(classification_report(y_test, y_predict))
```

```
             precision    recall  f1-score   support

        0.0       0.00      0.00      0.00        48
        1.0       0.58      1.00      0.73        66

avg / total       0.34      0.58      0.42       114
```

As we can see, our model did not do a good job in its predictions. It predicted that 48 healthy patients have cancer. We only achieved 34% accuracy!

Let’s explore ways to improve the performance of our model.

Improving our Model

The first process we will try is by **normalizing** our data

Data normalization is a feature scaling process that brings all values into range [0,1]

X’ = (X-X_min) / (X_max — X_min)

Normalize Training Data

In [43]:
```
X_train_min = X_train.min()
X_train_min
```

Out[43]:
```
mean radius                 6.981000
mean texture               10.380000
mean perimeter             43.790000
mean area                 143.500000
mean smoothness             0.052630
mean compactness            0.019380
mean concavity              0.000000
mean concave points         0.000000
mean symmetry               0.106000
mean fractal dimension      0.049960
radius error                0.111500
texture error               0.360200
perimeter error             0.757000
area error                  6.802000
smoothness error            0.001713
compactness error           0.002252
concavity error             0.000000
concave points error        0.000000
symmetry error              0.007882
fractal dimension error     0.000895
worst radius                7.930000
worst texture              12.490000
worst perimeter            50.410000
worst area                185.200000
worst smoothness            0.071170
worst compactness           0.027290
worst concavity             0.000000
```

In [25]:
```
X_train_max = X_train.max()
X_train_max
```

Out[25]:
```
mean radius                  28.11000
mean texture                 39.28000
mean perimeter              188.50000
mean area                  2501.00000
mean smoothness               0.14470
mean compactness              0.34540
mean concavity                0.42680
mean concave points           0.20120
mean symmetry                 0.30400
mean fractal dimension        0.09296
radius error                  2.87300
texture error                 4.88500
perimeter error              21.98000
area error                  542.20000
smoothness error              0.03113
compactness error             0.13540
concavity error               0.39600
concave points error          0.05279
symmetry error                0.07895
fractal dimension error       0.02984
worst radius                 36.04000
```

```
In [26]: X_train_range = (X_train_max- X_train_min)
         X_train_range
```

```
Out[26]: mean radius                 21.129000
         mean texture                28.900000
         mean perimeter             144.710000
         mean area                 2357.500000
         mean smoothness              0.092070
         mean compactness             0.326020
         mean concavity               0.426800
         mean concave points          0.201200
         mean symmetry                0.198000
         mean fractal dimension       0.043000
         radius error                 2.761500
         texture error                4.524800
         perimeter error             21.223000
         area error                 535.398000
         smoothness error             0.029417
         compactness error            0.133148
         concavity error              0.396000
         concave points error         0.052790
         symmetry error               0.071068
         fractal dimension error      0.028945
         worst radius                28.110000
         worst texture               37.050000
         worst perimeter            200.790000
         worst area                4068.800000
         worst smoothness             0.151430
         worst compactness            1.030710
```

```
In [27]: X_train_scaled = (X_train - X_train_min)/(X_train_range)
         X_train_scaled.head()
```

Out[27]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	wo ar
412	0.114345	0.391003	0.110290	0.053150	0.293907	0.126219	0.087512	0.025487	0.108081	0.401860	...	0.072394	0.418354	0.080681	0.028
461	0.967343	0.549827	0.988943	1.000000	0.605735	0.550334	0.851687	0.839463	0.505556	0.145814	...	1.000000	0.509582	1.000000	1.000
532	0.317052	0.205882	0.303849	0.183245	0.435973	0.163088	0.041050	0.093439	0.288384	0.269535	...	0.281750	0.208097	0.254943	0.144
495	0.373373	0.340138	0.361620	0.227953	0.469643	0.196522	0.159888	0.246074	0.215657	0.174884	...	0.287442	0.431579	0.266398	0.147
13	0.419755	0.469550	0.414000	0.271135	0.340828	0.247899	0.232849	0.266600	0.397475	0.079535	...	0.316969	0.409447	0.306738	0.169

5 rows × 30 columns

Normalize Training Data

```
In [28]: X_test_min = X_test.min()
         X_test_range = (X_test - X_test_min).max()
         X_test_scaled = (X_test - X_test_min)/X_test_range
```

Now, let's train our SVM model with our scaled (Normalized) datasets.

```
In [29]: svc_model = SVC()
         svc_model.fit(X_train_scaled, y_train)
```

```
Out[29]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)
```

Prediction with Scaled dataset

```
In [30]: y_predict = svc_model.predict(X_test_scaled)
         cm = confusion_matrix(y_test, y_predict)
```

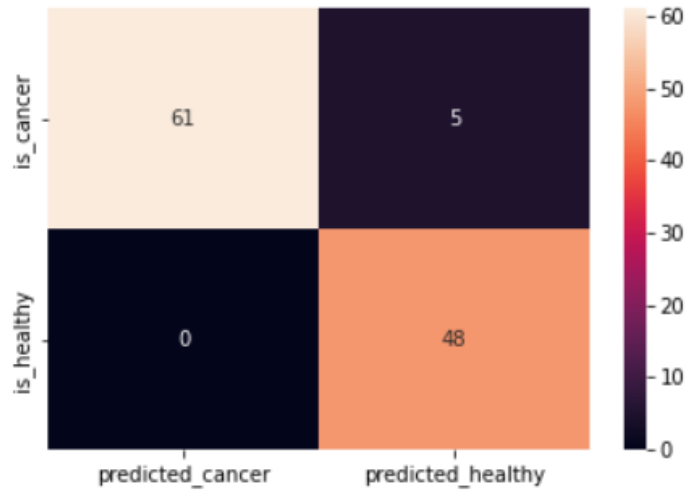
Confusion Matrix on Scaled dataset

```
[ ]  cm = np.array(confusion_matrix(y_test, y_predict, labels=[1,0]))
     confusion = pd.DataFrame(cm, index=['is_cancer', 'is_healthy'],
                              columns=['predicted_cancer','predicted_healthy'])
     confusion
```

	predicted_cancer	predicted_healthy
is_cancer	61	5
is_healthy	0	48

```
sns.heatmap(confusion,annot=True,fmt="d")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9620a820d0>
```



```
[ ]  print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
0.0	0.91	1.00	0.95	48
1.0	1.00	0.92	0.96	66
accuracy			0.96	114
macro avg	0.95	0.96	0.96	114
weighted avg	0.96	0.96	0.96	114

Our prediction got a lot better with only 0 false prediction(Predicted cancer instead of healthy). We achieved 99% accuracy!

CONCLUSION:

This project took us through the journey of explaining what “modeling” 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.

REFERENCES

http://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28diagnostic%29

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