Building a Simple Machine Learning Model on Breast Cancer Data

vishabh goel Follow
Sep 29, 2018 · 10 min read

Fig: Machine Learning Model

Introduction

Breast cancer (BC) is one of the most common cancers among women worldwide, representing the majority of new cancer cases and cancer-related deaths according to global statistics, making it a significant public health problem in today's society.

The early diagnosis of BC can improve the prognosis and chance of survival significantly, as it can promote timely clinical treatment to patients. Further accurate classification of benign tumors can prevent patients undergoing unnecessary treatments. Thus, the correct diagnosis of BC and classification of patients into malignant or benign groups is the subject of much research. Because of its unique advantages in critical features detection from complex BC datasets, machine learning (ML) is widely recognized as the methodology of choice in BC pattern classification and forecast modelling.

Classification and data mining methods are an effective way to classify data. Especially in medical field, where those methods are widely used in diagnosis and analysis to make decisions.

Recommended Screening Guidelines:

Mammography. The most important screening test for breast cancer is the mammogram. A mammogram is an X-ray of the breast. It can detect breast cancer up to two years before the tumor can be felt by you or your doctor.

Women age 40–45 or older who are at average risk of breast cancer should have a mammogram once a year.

Women at high risk should have yearly mammograms along with an MRI starting at age 30.

Some Risk Factors for Breast Cancer

The following are some of the known risk factors for breast cancer. However, most cases of breast cancer cannot be linked to a specific cause. Talk to your doctor about your specific risk.

Age. The chance of getting breast cancer increases as women age. Nearly 80 percent of breast cancers are found in women over the age of 50.

Personal history of breast cancer. A woman who has had breast cancer in one breast is at an increased risk of developing cancer in her other breast.

Family history of breast cancer. A woman has a higher risk of breast cancer if her mother, sister or daughter had breast cancer, especially at a young age (before 40). Having other relatives with breast cancer may also raise the risk.

Genetic factors. Women with certain genetic mutations, including changes to the BRCA1 and BRCA2 genes, are at higher risk of developing breast cancer during their lifetime. Other gene changes may raise breast cancer risk as well.

Childbearing and menstrual history. The older a woman is when she has her first child, the greater her risk of breast cancer. Also at higher risk are:

- Women who menstruate for the first time at an early age (before 12)
- Women who go through menopause late (after age 55)
- Women who've never had children

Phase 0—Data Preparation

We will use the UCI Machine Learning Repository for breast cancer dataset.

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

The dataset used in this story is publicly available and was created by Dr. William H. Wolberg, physician at the University Of Wisconsin Hospital at Madison, Wisconsin, USA. To create the dataset Dr. Wolberg used fluid samples, taken from patients with solid breast masses and an easy-to-use graphical computer program called Xcyt, which is capable of perform the analysis of cytological features based on a digital scan. The program uses a curve-fitting algorithm, to compute ten features from each one of the cells in the sample, than it calculates the mean value, extreme value and standard error of each feature for the image, returning a 30 real-valuated vector

Attribute Information:

1. ID number 2) Diagnosis (M = malignant, B = benign) 3–32)

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)

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.

Objectives

This analysis aims to observe which features are most helpful in predicting malignant or benign cancer and to see general trends that may aid us in model selection and hyper parameter selection. The goal is to classify whether the breast cancer is benign or malignant. To achieve this i have used machine learning classification methods to fit a function that can predict the discrete class of new input.

Phase 1—Data Exploration

We will be using ***Spyder*** to work on this dataset. We will first go with importing the necessary libraries and import our dataset to Spyder :

. . .

```
#importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
#importing our cancer dataset
dataset = pd.read_csv('cancer.csv')
X = dataset.iloc[:, 1:31].values
Y = dataset.iloc[:, 31].values
```

Fig : Dataset and X set after importing the dataset

We can examine the data set using the pandas' **head()** method.

```
dataset.head()
```

	id	radius_mean	...	fractal_dimension_worst	diagnosis
0	842302	17.99	...	0.11890	M
1	842517	20.57	...	0.08902	M
2	84300903	19.69	...	0.08758	M
3	84348301	11.42	...	0.17300	M
4	84358402	20.29	...	0.07678	M

Fig : top 5 data of our dataset

We can find the dimensions of the data set using the panda dataset 'shape' attribute.

```
print("Cancer data set dimensions :
{}".format(dataset.shape))
```

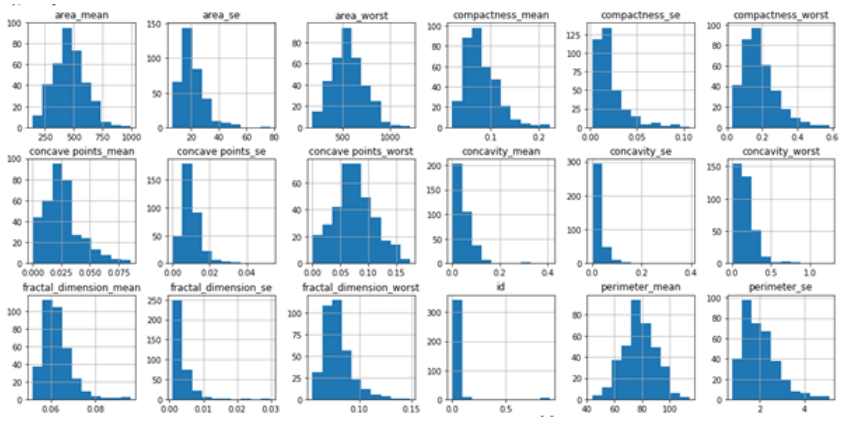
```
Cancer data set dimensions : (569, 32)
```

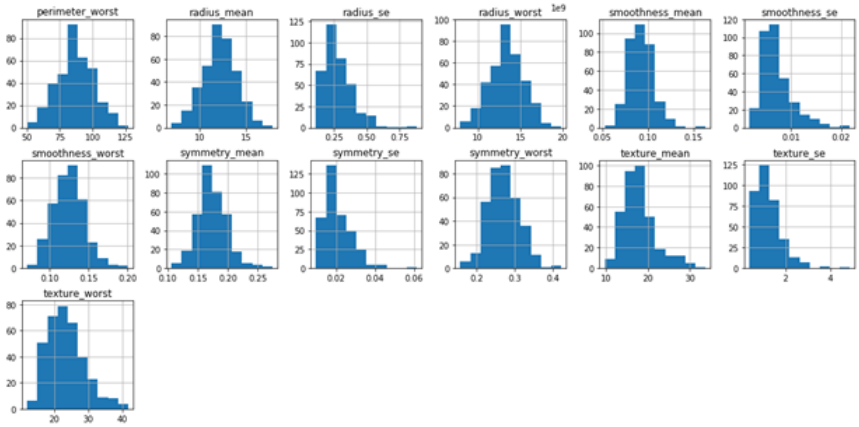
We can observe that the data set contain 569 rows and 32 columns. '*Diagnosis*' is the column which we are going to predict , which says if the cancer is M = malignant or B = benign. 1 means the cancer is malignant and 0 means benign. We can identify that out of the 569 persons, 357 are labeled as B (benign) and 212 as M (malignant).

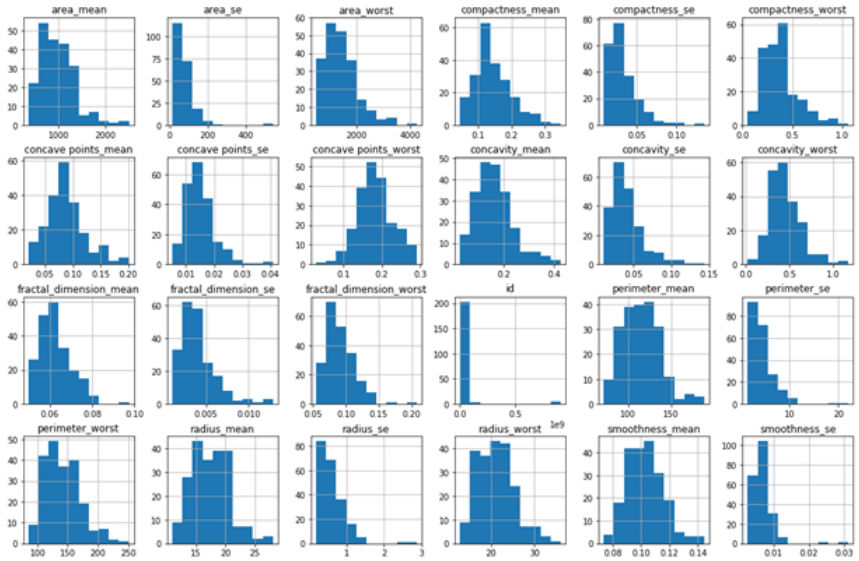
```
diagnosis
B    357
M    212
dtype: int64
```

Visualization of data is an imperative aspect of data science. It helps to understand data and also to explain the data to another person. Python has several interesting visualization libraries such as Matplotlib, Seaborn etc.

In this tutorial we will use pandas' visualization which is built on top of matplotlib, to find the data distribution of the features.


area_mean
area_se
area_worst
compactness_mean
compactness_se
compactness_worst
concave points_mean
concave points_se
concave points_worst
concavity_mean
concavity_se
concavity_worst
fractal_dimension_mean
fractal_dimension_se
fractal_dimension_worst
id
perimeter_mean
perimeter_se


perimeter_worst
radius_mean
radius_se
radius_worst
1e9
smoothness_mean
smoothness_se
smoothness_worst
symmetry_mean
symmetry_se
symmetry_worst
texture_mean
texture_se
texture_worst


area_mean
area_se
area_worst
compactness_mean
compactness_se
compactness_worst
concave points_mean
concave points_se
concave points_worst
concavity_mean
concavity_se
concavity_worst
fractal_dimension_mean
fractal_dimension_se
fractal_dimension_worst
id
perimeter_mean
perimeter_se
perimeter_worst
radius_mean
radius_se
radius_worst
1e9
smoothness_mean
smoothness_se

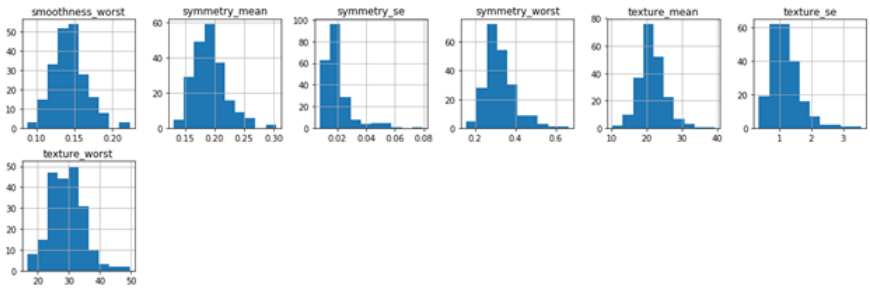


Fig : Visualization of Dataset

Missing or Null Data points

We can find any missing or null data points of the data set (if there is any) using the following pandas function.

```
dataset.isnull().sum()
dataset.isna().sum()
```

```
id                        0
radius_mean               0
texture_mean              0
perimeter_mean            0
area_mean                 0
smoothness_mean           0
compactness_mean          0
concavity_mean            0
concave points_mean       0
symmetry_mean             0
fractal_dimension_mean    0
radius_se                 0
texture_se                0
perimeter_se              0
area_se                   0
smoothness_se             0
compactness_se            0
concavity_se              0
concave points_se         0
symmetry_se               0
fractal_dimension_se      0
radius_worst              0
texture_worst             0
perimeter_worst           0
area_worst                0
smoothness_worst          0
compactness_worst         0
concavity_worst           0
concave points_worst      0
symmetry_worst            0
fractal_dimension_worst   0
diagnosis                 0
dtype: int64
```

Fig : Observe missing data

Phase 2—Categorical Data

Categorical data are variables that contain label values rather than numeric values.The number of possible values is often limited to a fixed set.

For example, users are typically described by country, gender, age group etc.

We will use Label Encoder to label the categorical data. Label Encoder is the part of SciKit Learn library in Python and used to convert categorical data, or text data, into numbers, which our predictive models can better understand.

```
#Encoding categorical data values
from sklearn.preprocessing import LabelEncoder
labelencoder_Y = LabelEncoder()
Y = labelencoder_Y.fit_transform(Y)
```

Index	0
0	M
1	M
2	M
3	M
4	M
5	M
6	M
7	M
8	M
9	M
10	M
11	M
12	M
13	M
14	M

Fig: Diagnosis Data without Encoding

	0
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1

Fig: Diagnosis Data after Encoding

Splitting the dataset

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model's prediction on this subset.

We will do this using SciKit-Learn library in Python using the train_test_split method.

```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size = 0.25, random_state = 0)
```

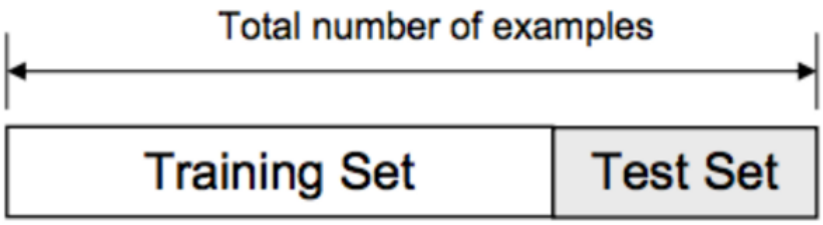


Fig: Training and test set

Phase 3—Feature Scaling

Most of the times, your dataset will contain features highly varying in magnitudes, units and range. But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations. We need to bring all features to the same level of magnitudes. This can be achieved by scaling. This means that you're transforming your data so that it fits within a specific scale, like 0–100 or 0–1.

We will use StandardScaler method from SciKit-Learn library.

```
#Feature Scaling
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Phase 4—Model Selection

This is the most exciting phase in Applying Machine Learning to any Dataset. It is also known as Algorithm selection for Predicting the best results.

Usually Data Scientists use different kinds of Machine Learning algorithms to the large data sets. But, at high level all those different algorithms can be classified in two groups : supervised learning and unsupervised learning.

Without wasting much time, I would just give a brief overview about these two types of learnings.

Supervised learning : Supervised learning is a type of system in which both input and desired output data are provided. Input and output data are labelled for classification to provide a learning basis for future data processing. Supervised learning problems can be further grouped into **Regression** and **Classification** problems.

A **regression** problem is when the output variable is a real or continuous value, such as “salary” or “weight”.

A **classification** problem is when the output variable is a category like filtering emails “spam” or “not spam”

Unsupervised Learning : Unsupervised learning is the algorithm using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.

In our dataset we have the outcome variable or Dependent variable i.e Y having only two set of values, either M (Malign) or B(Benign). So we will use Classification algorithm of supervised learning.

We have different types of classification algorithms in Machine Learning :-

1. Logistic Regression

2. Nearest Neighbor

3. Support Vector Machines

4. Kernel SVM

5. Naïve Bayes

6. Decision Tree Algorithm

7. Random Forest Classification

Lets start applying the algorithms :

We will use sklearn library to import all the methods of classification algorithms.

We will use LogisticRegression method of model selection to use Logistic Regression Algorithm,

```
#Using Logistic Regression Algorithm to the Training Set
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, Y_train)
```

```
#Using KNeighborsClassifier Method of neighbors class to use
Nearest Neighbor algorithm
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5, metric =
'minkowski', p = 2)
classifier.fit(X_train, Y_train)
```

```
#Using SVC method of svm class to use Support Vector Machine
Algorithm
```

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, Y_train)
```

```
#Using SVC method of svm class to use Kernel SVM Algorithm
```

```
from sklearn.svm import SVC
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, Y_train)
```

```
#Using GaussianNB method of naïve_bayes class to use Naïve
Bayes Algorithm
```

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, Y_train)
```

```
#Using DecisionTreeClassifier of tree class to use Decision
Tree Algorithm
```

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy',
random_state = 0)
classifier.fit(X_train, Y_train)
```

```
#Using RandomForestClassifier method of ensemble class to
use Random Forest Classification algorithm
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
classifier = RandomForestClassifier(n_estimators = 10,
criterion = 'entropy', random_state = 0)
classifier.fit(X_train, Y_train)
```

We will now predict the test set results and check the accuracy with each of our model:

```
Y_pred = classifier.predict(X_test)
```

To check the accuracy we need to import confusion_matrix method of metrics class. The confusion matrix is a way of tabulating the number of mis-classifications, i.e., the number of predicted classes which ended up in a wrong classification bin based on the true classes.

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
```

We will use Classification Accuracy method to find the accuracy of our models. Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

Fig: Accuracy

To check the correct prediction we have to check confusion matrix object and add the predicted results diagonally which will be number of correct prediction and then divide by total number of predictions.

	0	1
0	87	3
1	3	50

Fig: Confusion Matrix

After applying the different classification models, we have got below accuracies with different models:

1. Logistic Regression—95.8%

2. Nearest Neighbor—95.1%

3. Support Vector Machines—97.2%

4. Kernel SVM—96.5%

5. Naive Bayes—91.6%

6. Decision Tree Algorithm—95.8%

7. Random Forest Classification—98.6%

So finally we have built our classification model and we can see that Random Forest Classification algorithm gives the best results for our dataset. Well its not always applicable to every dataset. To choose our model we always need to analyze our dataset and then apply our machine learning model.

This is a basic application of Machine Learning Model to any dataset. Feel free to ask questions if you have any doubts. Drop an email to: vishabh1010@gmail.com or contact me through linked-in.

You can find the code on github and try it on Ipython console.

Remember to always keep in mind the problem of overfitting and underfitting as well.

Hope you enjoyed the article…..