Documentation of -

Predict-Diagnosis-of-a-Breast-Tumor-Using-Machine-Learning

By

Rakin Mohammad Sifullah

Abstract:

Breast cancer is the most prevalent cancer in women, accounting for almost one-third of all cancer diagnoses in this population. It is also the second most significant cause of cancer-related death in women. Breast tissue cells grow abnormally, creating what is known as a tumor, which leads to breast cancer. Tumors are not always indicative of cancer; they may be benign, pre-malignant, or malignant (cancerous). Breast cancer is frequently diagnosed with procedures like MRI, mammography, ultrasound, and biopsy.

Given the findings of a breast fine-needle aspiration (FNA) test, which uses a thin needle akin to the one used for blood samples to take some fluid or cells from a breast lesion or cyst (a lump, sore, or swelling). Because of this, I built a model that uses two training classes to categorize breast cancer tumors: here 1 indicates Malignant (Cancerous) is present and 0 means Bening (Not Cancerous) is Absent.

The prediction splits into two categories because the labels in the data are discrete (i.e. Malignant or benign). This is a classification issue in terms of machine learning.

Therefore, the objective is to categorize whether breast cancer is benign or malignant and to forecast the recurrence and non-recurrence of malignant instances over time. To do this, we fitted a function that can predict the discrete class of fresh input using machine learning classification algorithms.

Data Sources

The **Breast Cancer** datasets are available as a machine learning repository maintained by the University of California, Irvine. The dataset contains **569 samples of malignant and benign tumor cells**.

- The first two columns in the dataset store the unique ID numbers of the samples and the corresponding diagnosis (M = malignant, B = benign), respectively.
- Columns 3-32 contain 30 real-value features that have been computed from digitized images of the cell nuclei, which can be used to build a model to predict whether a tumor is benign or malignant.

Load Dataset

First, load the supplied CSV file using additional options in the Pandas **read_csv** function.

Inspecting the data

The first step is to visually inspect the new data set. There are multiple ways to achieve this:

- The easiest is to request the first few records using the DataFrame **data.head()** method. By default, **data.head()** returns the first 5 rows from the DataFrame object df (excluding the header row).
- Alternatively, one can also use **df.tail()** to return the five rows of the data frame.
- For both head and tail methods, there is an option to specify the number of records by including the required number in between the parentheses when calling either method. Inspecting the data

Exploratory Data Analysis

Now that we have a good intuitive sense of the data, the Next step involves taking a closer look at attributes and data values. In this section, I am getting familiar with the data, which will provide useful knowledge for data pre-processing.

2.1 Objectives of Data Exploration

Exploratory data analysis (EDA) is a very important step that takes place after feature engineering and acquiring data and it should be done before any modeling. This is because it is very important for a data scientist to be able to understand the nature of the data without making assumptions. The results of data exploration can be extremely useful in grasping the structure of the data, the distribution of the values, and the presence of extreme values and interrelationships within the data set.

The purpose of EDA is:

- to use summary statistics and visualizations to better understand data, *find clues about the tendencies of the data, its quality and to formulate assumptions and the hypothesis of our analysis
- For data preprocessing to be successful, it is essential to have an overall picture of your data Basic statistical descriptions can be used to identify properties of the data and highlight which data values should be treated as noise or outliers.**

Next step is to explore the data. There are two approached used to examine the data using:

1. ***Descriptive statistics*** is the process of condensing key characteristics of the data set into simple numeric metrics. Some of the common metrics used are mean, standard deviation, and correlation.
2. ***Visualization*** is the process of projecting the data, or parts of it, into Cartesian space or into abstract images. In the data mining process, data exploration is leveraged in many different steps including preprocessing, modeling, and interpretation of results.

Unimodal Data Visualizations

One of the main goals of visualizing the data here is to observe which features are most helpful in predicting malignant or benign cancer. The other is to see general trends that may aid us in model selection and hyper-parameter selection.

Apply 3 techniques that you can use to understand each attribute of your dataset independently.

- Histograms.
- Density Plots.
- Box and Whisker Plots.

Visualize distribution of data via histograms

Histograms are commonly used to visualize numerical variables. A histogram is similar to a bar graph after the values of the variable are grouped (binned) into a finite number of intervals (bins).

Histograms group data into bins and provide you with a count of the number of observations in each bin. From the shape of the bins, you can quickly get a feeling of whether an attribute is Gaussian, skewed, or even has an exponential distribution. It can also help you see possible outliers.

Observation

> We can see that perhaps the attributes **concavity** and **concavity_point** may have an exponential distribution ( ). We can also see that perhaps the texture and smooth and symmetry attributes may have a Gaussian or nearly Gaussian distribution. This is interesting because many machine learning techniques assume a Gaussian univariate distribution on the input variables.

Visualize the distribution of data via density plots

We can see that perhaps the attributes perimeter, radius, area, concavity, and compactness may have an exponential distribution( ). We can also see that perhaps the texture and smooth and

symmetry attributes may have a Gaussian or nearly Gaussian distribution. This is interesting because many machine learning techniques assume a Gaussian univariate distribution on the input variables.

Visualize the distribution of data via box plots

We can see that perhaps the attributes perimeter, radius, area, concavity,ompactness may have an exponential distribution( ). We can also see that perhaps the texture and smooth and symmetry attributes may have a Gaussian or nearly Gaussian distribution. This is interesting because many machine learning techniques assume a Gaussian univariate distribution on the input variables.

Multimodal Data Visualizations

- Scatter plots
- Correlation matrix

Observation:
We can see a strong positive relationship exists with mean values parameters between **1** to **0.75**.

- The mean area of the tissue nucleus has a strong positive correlation with mean values of radius and parameter;
- Some parameters are moderately positive correlated (r between 0.5-0.75)are concavity and area, concavity and perimeter, etc
- Likewise, we see some strong negative correlation between fractal_dimension with radius, texture, parameter mean values.

Summary:

- Mean values of cell radius, perimeter, area, compactness, concavity, and concave points can be used in the classification of cancer. Larger values of these parameters tend to show a correlation with malignant tumors.
- mean values of texture, smoothness, symmetry, or factual 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 warrant further cleanup.

Pre-Processing the data

Introduction

Data preprocessing is a crucial step for any data analysis problem. It is often a very good idea to prepare your data in such a way to best expose the structure of the problem to the machine learning algorithms that you intend to use. This involves a number of activities such as:

- Assigning numerical values to categorical data;
- Handling missing values; and
- Normalizing the features (so that features on small scales do not dominate when fitting a model to the data).

In Part_2, I explored the data, to help gain insight into the distribution of the data as well as how the attributes correlate to each other. I identified some features of interest. In this notebook, I use feature selection to reduce high-dimension data, and feature extraction, and transformation for dimensionality reduction.

Goal:

Find the most predictive features of the data and filter it so it will enhance the predictive power of the analytics model.

Label encoding

Here, I assign the 30 features to a NumPy array X, and transform the class labels from their original string representation (M and B) into integers

After encoding the class labels(diagnosis) in an array y, the malignant tumors are now represented as class 1(i.e presence of cancer cells) and the benign tumors are represented as class 0 (i.e no cancer cells detection), respectively, illustrated by calling the transform method of LabelEncorder on two dummy variables.**

Assessing Model Accuracy: Split data into training and test sets

The simplest method to evaluate the performance of a machine learning algorithm is to use different training and testing datasets. Here I will

- Split the available data into a training set and a testing set. (70% training, 30% test)
- Train the algorithm on the first part,
- make predictions on the second part and
- evaluate the predictions against the expected results.

The size of the split can depend on the size and specifics of your dataset, although it is common to use 67% of the data for training and the remaining 33% for testing.

Feature Standardization

- Standardization is a useful technique to transform attributes with a Gaussian distribution and differing means and standard deviations to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1.
- As seen in **Part_2** the raw data has differing distributions which may have an impact on the most ML algorithms. Most machine learning and optimization algorithms behave much better if features are on the same scale.

Let's evaluate the same algorithms with a standardized copy of the dataset. Here, I use sklearn to scale and transform the data such that each attribute has a mean value of zero and a standard deviation of one

Feature decomposition using Principal Component Analysis (PCA)

From the pair plot in **Part_2**, a lot of feature pairs divide nicely the data to a similar extent, therefore, it makes sense to use one of the dimensionality reduction methods to try to use as many features as possible and maintain as much information as possible when working with only 2 dimensions. I will use PCA.

Now, what we got after applying the linear PCA transformation is a lower-dimensional subspace (from 3D to 2D in this case), where the samples are **most spread** along the new feature axes.

Summary: Data Preprocessing Approach used

1. assign features to a NumPy array X, and transform the class labels from their original string representation (M and B) into integers
2. Split data into training and test set
3. Standardize the data.
4. Obtain the Eigenvectors and Eigenvalues from the covariance matrix or correlation matrix
5. Sort eigenvalues in descending order and choose the kk eigenvectors that correspond to the kk largest eigenvalues where k is the number of dimensions of the new feature subspace

6. (k≤dk≤d)
7. .
8. Construct the projection matrix W from the selected k eigenvectors.
9. Transform the original dataset X via W to obtain a k-dimensional feature subspace Y.

It is common to select a subset of features that have the largest correlation with the class labels. The effect of feature selection must be assessed within a complete modeling pipeline in order to give you an unbiased estimate of your model's true performance. Hence, in the next section, you will first be introduced to cross-validation, before applying the PCA-based feature selection strategy in the model building pipeline.

A predictive model using Support Vector Machine (SVM)

Support vector machines (SVMs) learning algorithm will be used to build the predictive model. SVMs are one of the most popular classification algorithms and have an elegant way of transforming nonlinear data so that one can use a linear algorithm to fit a linear model to the data (Cortes and Vapnik 1995).

Kernelized support vector machines are powerful models and perform well on a variety of datasets.

1. SVMs allow for complex decision boundaries, even if the data has only a few features.
2. They work well on low-dimensional and high-dimensional data (i.e., few and many features), but don’t scale very well with the number of samples.
   Running an SVM on data with up to 10,000 samples might work well, but working with datasets of size 100,000 or more can become challenging in terms of runtime and memory usage.
3. SVMs require careful preprocessing of the data and tuning of the parameters. This is why, these days, most people instead use tree-based models such as random forests or gradient boosting (which require little or no preprocessing) in many applications.
4. SVM models are hard to inspect; it can be difficult to understand why a particular prediction was made, and it might be tricky to explain the model to a nonexpert.

Important Parameters

The important parameters in kernel SVMs are the

- Regularization parameter C,
- The choice of the kernel,(linear, radial basis function(RBF) or polynomial)
- Kernel-specific parameters.

gamma and C both control the complexity of the model, with large values in either resulting in a more complex model. Therefore, good settings for the two parameters are usually strongly correlated, and C and gamma should be adjusted together.

Classification with cross-validation

As discussed in **Part_3** splitting the data into test and training sets is crucial to avoid overfitting. This allows the generalization of real, previously-unseen data. Cross-validation extends this idea further. Instead of having a single train/test split, we specify **so-called folds** so that the data is divided into similarly-sized folds.

- Training occurs by taking all folds except one – referred to as the holdout sample.
- On the completion of the training, you test the performance of your fitted model using the holdout sample.
- The holdout sample is then thrown back with the rest of the other folds, and a different fold is pulled out as the new holdout sample.
- Training is repeated again with the remaining folds and we measure performance using the holdout sample. This process is repeated until each fold has had a chance to be a test or holdout sample.
- The expected performance of the classifier, called cross-validation error, is then simply an average of error rates computed on each holdout sample.

This process is demonstrated by first performing a standard train/test split, and then computing cross-validation error.

Model Accuracy: Receiver Operating Characteristic (ROC) curve

In statistical modeling and machine learning, a commonly-reported performance measure of model accuracy for binary classification problems is Area Under the Curve (AUC).

To understand what information the ROC curve conveys, consider the so-called confusion matrix that essentially is a two-dimensional table where the classifier model is on one axis (vertical), and

ground truth is on the other (horizontal) axis, as shown below. Either of these axes can take two values (as depicted)

Model says "+"	**Model says "-"**	
True positive	False negative	**Actual: "+"**
False positive	True negative	Actual: "-"

In a ROC curve, you plot “True Positive Rate” on the Y-axis and “False Positive Rate” on the X-axis, where the values “true positive”, “false negative”, “false positive”, and “true negative” are events (or their probabilities) as described above. The rates are defined according to the following:

- True positive rate (or sensitivity)}: tpr = tp / (tp + fn)
- False positive rate: fpr = fp / (fp + tn)
- True negative rate (or specificity): tnr = tn / (fp + tn)

In all definitions, the denominator is a row margin in the above confusion matrix. Thus, one can express

- the true positive rate (tpr) as the probability that the model says "+" when the real value is true "+" (i.e., a conditional probability). However, this does not tell you how likely you are to be correct when calling "+" (i.e., the probability of a true positive, conditioned on the test result is "+").

Observation

There are two possible predicted classes: "1" and "0". Malignant = 1 (indicates presence of cancer cells) and Benign = 0 (indicates absence).

- The classifier made a total of 174 predictions (i.e 174 patients were being tested for the presence of breast cancer).
- Out of those 174 cases, the classifier predicted "yes" 58 times, and "no" 113 times.
- In reality, 64 patients in the sample have the disease, and 107 patients do not.

Rates as computed from the confusion matrix

1. **Accuracy**: Overall, how often is the classifier correct?
    a. (TP+TN)/total = (57+106)/171 = 0.95
2. **Misclassification Rate**: Overall, how often is it wrong?
    a. (FP+FN)/total = (1+7)/171 = 0.05 equivalent to 1 minus Accuracy also known as ***"Error Rate"***
3. **True Positive Rate:** When it's actually yes, how often does it predict 1?
    a. TP/actual yes = 57/64 = 0.89 also known as "Sensitivity" or ***"Recall"***
4. **False Positive Rate**: When it's actually 0, how often does it predict 1?
    a. FP/actual no = 1/107 = 0.01
5. **Specificity**: When it's actually 0, how often does it predict 0? also know as **true positive rate**
    a. TN/actual no = 106/107 = 0.99 equivalent to 1 minus False Positive Rate
6. **Precision**: When it predicts 1, how often is it correct?
    a. TP/predicted yes = 57/58 = 0.98
7. **Prevalence**: How often does the yes condition actually occur in our sample?
    a. actual yes/total = 64/171 = 0.34

- To interpret the ROC correctly, consider what the points that lie along the diagonal represent. For these situations, there is an equal chance of "+" and "-" happening. Therefore, this is not that different from making a prediction by tossing of an unbiased coin. Put simply, the classification model is random.
- For the points above the diagonal, tpr > fpr, and the model says that you are in a zone where you are performing better than random. For example, assume tpr = 0.99 and fpr = 0.01, Then, the probability of being in the true positive group is $(0.99/(0.99+0.01))=99\%$. Furthermore, holding fpr constant, it is easy to see that the more vertically above the diagonal you are positioned, the better the classification model.

Optimizing the SVM Classifier

Machine learning models are parameterized so that their behavior can be tuned for a given problem. Models can have many parameters and finding the best combination of parameters can be treated as a search problem. In this part, we'll aim to tune the parameters of the SVM Classification model using scikit-learn.

```
The classifier accuracy score is 0.96

              precision    recall  f1-score   support

           0       0.95      0.99      0.97       107
           1       0.98      0.91      0.94        64

    accuracy                           0.96       171
   macro avg       0.96      0.95      0.96       171
weighted avg       0.96      0.96      0.96       171
```

Importance of optimizing a classifier

We can tune two key parameters of the SVM algorithm:

- the value of C (how much to relax the margin)
- and the type of kernel.

The default for SVM (the SVC class) is to use the Radial Basis Function (RBF) kernel with a C value set to 1.0. Like with KNN, we will perform a grid search using 10-fold cross-validation with a standardized copy of the training dataset. We will try a number of simpler kernel types and C values with less bias and more bias (less than and more than 1.0 respectively).

Python scikit-learn provides two simple methods for algorithm parameter tuning:

- Grid Search Parameter Tuning.
- Random Search Parameter Tuning.

Conclusion

This work demonstrates the modeling of breast cancer as a classification task using a Support Vector Machine

The SVM performs better when the dataset is standardized so that all attributes have a mean value of zero and a standard deviation of one. We can calculate this from the entire training dataset and apply the same transform to the input attributes from the validation dataset.

Automate the ML process using pipelines

There are standard workflows in a machine learning project that can be automated. In Python scikit-learn, Pipelines help to clearly define and automate these workflows.

- Pipelines help overcome common problems like data leakage in your test harness.
- Python scikit-learn provides a Pipeline utility to help automate machine learning workflows.
- Pipelines work by allowing for a linear sequence of data transforms to be chained together culminating in a modeling process that can be evaluated.

Evaluate Some Algorithms

Now it is time to create some models of the data and estimate their accuracy on unseen data. Here is what we are going to cover in this step:

1. Separate out a validation dataset.
2. Setup the test harness to use 10-fold cross-validation.
3. Build 5 different models
4. Select the best model

Tuning the hyper-parameters: k-NN hyperparameters

For your standard k-NN implementation, there are two primary hyperparameters that you'll want to tune:

- The number of neighbors k.
- The distance metric/similarity function.

Both of these values can dramatically affect the accuracy of your k-NN classifier. Grid object is ready to do 10-fold cross-validation on a KNN model using classification accuracy as the evaluation metric In addition, there is a parameter grid to repeat the 10-fold cross-validation process 30 times Each time, the n_neighbors parameter should be given a different value from the list We can't give **GridSearchCV** just a list We've to specify n_neighbors should take on 1 through 30 You can set **n_jobs = -1** to run computations in parallel (if supported by your computer and OS)

Summary

Worked through a classification predictive modeling machine learning problem from end-to-end using Python. Specifically, the steps covered were:

1. Problem Definition (Breast Cancer data).
2. Loading the Dataset.
3. Analyze Data (same scale but di↵erent distributions of data).
    - Evaluate Algorithms (KNN looked good).
    - Evaluate Algorithms with Standardization (KNN and SVM looked good).
4. Algorithm Tuning (K=19 for KNN was good, SVM with an RBF kernel and C=100 was best)..

5. Finalize Model (use all training data and confirm using validation dataset)