Supervised machine learning algorithms application on dataset with binomial outcome

Abstract:

In the following report I will be discussing several machine learning algorithms and their application on breast cancer dataset. Dependent variable is a diagnosis i.e. I would like to predict whether a patient has a malignant or benign tumor. I will be applying 5 different algorithms: Support vector machines, K nearest neighbor classification, Random forest Classifier, Decision Tree Classifier and Logistic Regression. I will be testing algorithms using K-fold clustering.

Introduction:

Data presented in this report is a breast cancer data (Wisconsin) that is available at UCI ML repository. I selected this data because breast cancer is a very serious and infamous disease that takes a lot of lives every year. I wanted to see which features were the most important in putting diagnosis of benign or malignant cancer. Data has 569 observation with 33 columns-one of them being diagnosis-dependent variable. I dropped two columns called “Unnamed:32” and “ID” because they are redundant. So, for our analysis I will use 30 features to predict diagnosis. There were no missing values found.

Exploratory data analysis:

As a first step of our exploratory data analysis I decide to plot number of malignant versus number of benign diagnosis I have. As you can see from the Figure 1, number of benign diagnosis is 357 and malignant -212. I find data to be balanced and well representative of a daily life data where more people have benign cancer.

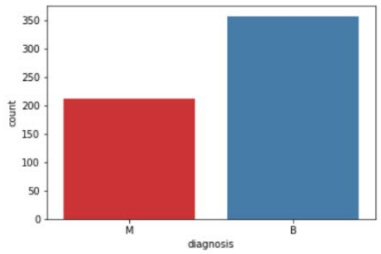


Figure 1:Histogram of diagnoses: M-Malignant and B-Benign

Later I decided to plot histograms of features I have and see how values of the features change depending on the diagnosis. For some of the features, like perimeter_mean, represented in the figure 2, benign and malignant cancer are well differentiated but for some features, like smoothness_mean- measures overlap and I cannot easily differentiate data just by looking at the histogram. I plotted histogram for all the features and realized that just plotting if them is not enough and I need statistical test in order to choose best features.

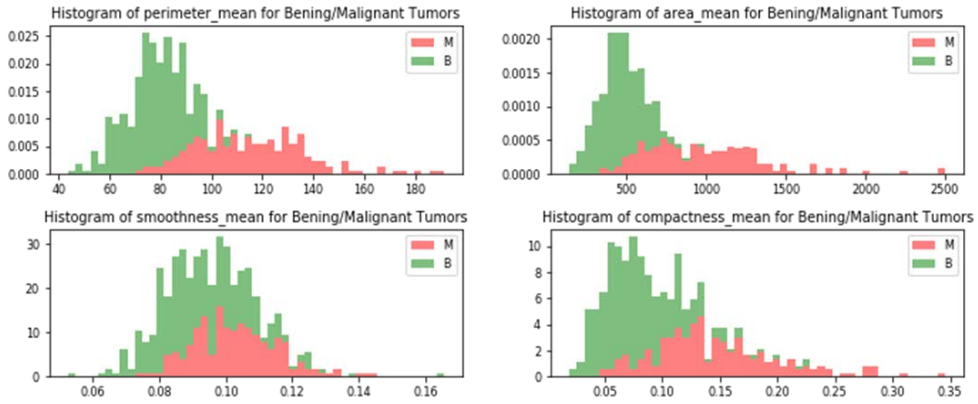


Figure 2:Histogram of 4/29 predictors divided by diagnosis

I also did a correlation heatmap in order to see if there is multicollinearity. In the figure below you can see just a small part of the correlation heatmap. If two variables are highly correlated, corresponding square will represent Pearson correlation and color from white-highly correlated to dark-not correlated. As I expected, area, perimeter, radius and all corresponding features are perfectly correlated and therefore the reis no need to use all of them But since there are a lot of features manually choosing one by one is time consuming so I will use statistical analysis in order to decide which features to include in the final model.

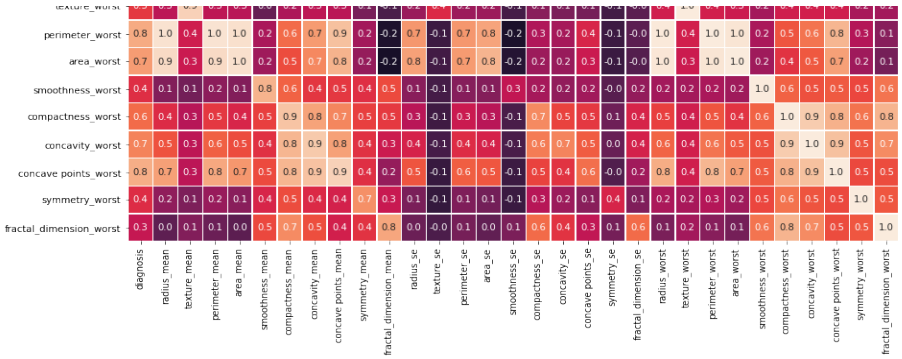


Figure 3: Small part of the correlation heatmap

Methodology

After exploring my dataset, I decided to implement some of the most popular supervised learning algorithms such as Support vector machines, K nearest neighbor classification, Random forest Classifier, Decision Tree Classifier and Logistic Regression.

For all algorithms firstly I train the data and then calculated accuracy of the model on that data. In order to be more objective, I will be testing algorithms using K-fold clustering, to be specific 4-fold clustering. I also defined shuffle=True, so that data is shuffled before divided into 4 folds. After running each algorithm, I also calculate average accuracy of the 4 Cross Validation folds.

1. SVM

First algorithm I implemented was Support Vector Machines (SVM). As we know, SVM is a discriminative classifier defined by a separating hyperplane i.e. given dependent variable, the algorithm outputs an optimal hyperplane which categorizes new examples. SVM algorithms use a set of mathematical functions that are defined as the kernel. There are several types of kernels exist but in my analysis I used only linear, radial basis function (RBF) and sigmoid.

Gaussian radial basis function (RBF) is a default Kernel in the python SVM library. It is a general-purpose kernel; used when there is no prior knowledge about the data and uses following formula:

$$k(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\gamma \|\mathbf{x_i} - \mathbf{x_j}\|^2) \text{where } \gamma > 0 \text{ or } \gamma = 1/2\sigma^2$$

In python, default value for gamma is 1 / (n_features * X.var()). I used this value of gamma for all SVM kernels. My training accuracy for SVM (kernel=rbf) is 100%, however for the 4-fold Cross validation, scores are 67%, 61%, 67%, 65% which makes average accuracy of 65%. Even though this score is low at this point I cannot make any conclusions since I didn't test other algorithms.

For **Linear Kernel** the equation for prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

f(x) = b(0) + sum(ai * (x,xi))

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients b0 and ai (for each input) must be estimated from the training data by the learning algorithm.

Applying this Kernel gave much better results. Even though accuracy is **96.6%**, CV scores are 98%, 93%, 90%, 96% making on average ***94.7%***. This means that Linear kernel performs much better and therefore is a candidate on best performing algorithm so far.

Lastly, **Sigmoid Kernel** which is similar to sigmoid function in logistic regression has following formula:

$$k(x, y) = \tanh(\alpha x^T y + c)$$

I didn't have any expectation prior, but the sigmoid kernel came up with the lowest scores. Accuracy score for training data is ***62%*** while CV scores are 62%, 59%, 64%, 64% making on average ***62%***. This the worst performing algorithms so far, where even training data accuracy score is low.

2. K-nearest Neighbor classifier

The KNN algorithm assumes that similar things exist in close proximity i.e. similar things are near to each other. In python, default number of neighbors is 5. I find this number pretty good taking into consideration the fact that we have only two groups: malignant and benign. There are two ways to calculate proximity of the observations to each other. Distance is calculated using Minkowski metric and when p=1it becomes Manhattan distance, when p=2 it is Euclidean distance and for other values of p it calculates Minkowski l_p distance. Figure below show the formulas used I order to calculated specifies distances.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

For the simplicity in my analysis I included Euclidean, Manhattan and Minkowski p=3 distance.

Euclidean distance is the default method used in the K-nearest Neighbors classifier. For Manhattan, p=1 should be used. Summary of the results for all three types of distances is presented in the table below:

Distance type	Train data accuracy	CV1	CV2	CV3	CV4	Average accuracy score
Manhattan	**95.4%**	94.4%	91.5%	92.3%	95.8%	**93.5%**
Euclidean	**94.7%**	93.0%	92.3%	91.5%	95.1%	**93.0%**
Minkowski L_3	**94.6%**	93.7%	92.3%	91.5%	95.1%	**93.1%**

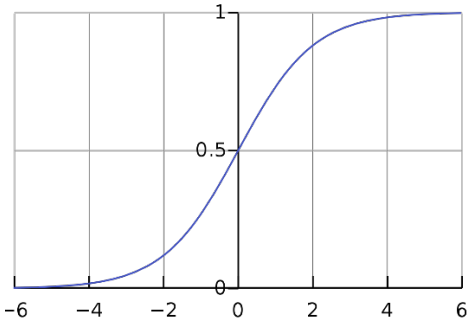
All distance calculation methods give similar and good results results but still lower than SVM linear kernel.

3. Logistic Regression

Logistic regression is also called the sigmoid It's an S-shaped curve that can takes a value and maps it into a value between 0 and 1.

$$S(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1}$$

where x is the value that you want to transform. Below is a plot of logistic curve.



In order to create less complex model when you have a large number of features in your dataset, some of the Regularization techniques are used to address over-fitting and feature selection. In Python, you can

decide what type of regularization you want to apply to your model. Default is L2 which is Ridge regression, other one is L1- Lasso regression.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Ridge regression:

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Lasso Regression:

The key difference between these two is the penalty term.

Logistic regression with ridge penalty

I first applied ridge regression penalty since it was the default one. It gave me the following results: Training data accuracy is ***95.9%***, while CV scores are 97.9%, 92.2%,92.9%, 95.7% making average accuracy score ***94.7%***. Penalty term decreased coefficient terms of all predictors but still all of them were still present in the model. This model gave similarly good results as the Linear kernel of SVM.

Logistic regression with lasso penalty

Prior running code I knew that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. This algorithm gave me following results: Training accuracy score is ***95.6%*** and CV scores are 98.6%, 92.9%, 92.2%, 95.7% making average testing accuracy of ***94.8%.*** Speaking of coefficients of predictors:

```
[-4.18191687, -0.06664095,  0.10488062,  0.02002074,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
  0.        , -0.93627889, -0.12181023,  0.11256291,  0.        ,
  0.        ,  0.        ,  0.        ,  0.        ,  0.        ,
 -0.15383812,  0.3068089 ,  0.18459758,  0.01245451,  0.        ,
  0.        ,  2.70045895,  0.        ,  0.        ,  0.        ]
```

We can see from the matrix above that Lasso penalty removed 18 predictors leaving only 12, which are: radius_mean,texture_mean,perimeter_mean,area_mean, texture_se, perimeter_se, area_se, radius_worst, texture_worst, perimeter_worst, area_worst, concavity_worst.

Since less predictors decrease running time and accuracy score are pretty good at this time, I find this algorithm to be the most optimal one.

4. Decision Tree Classifier

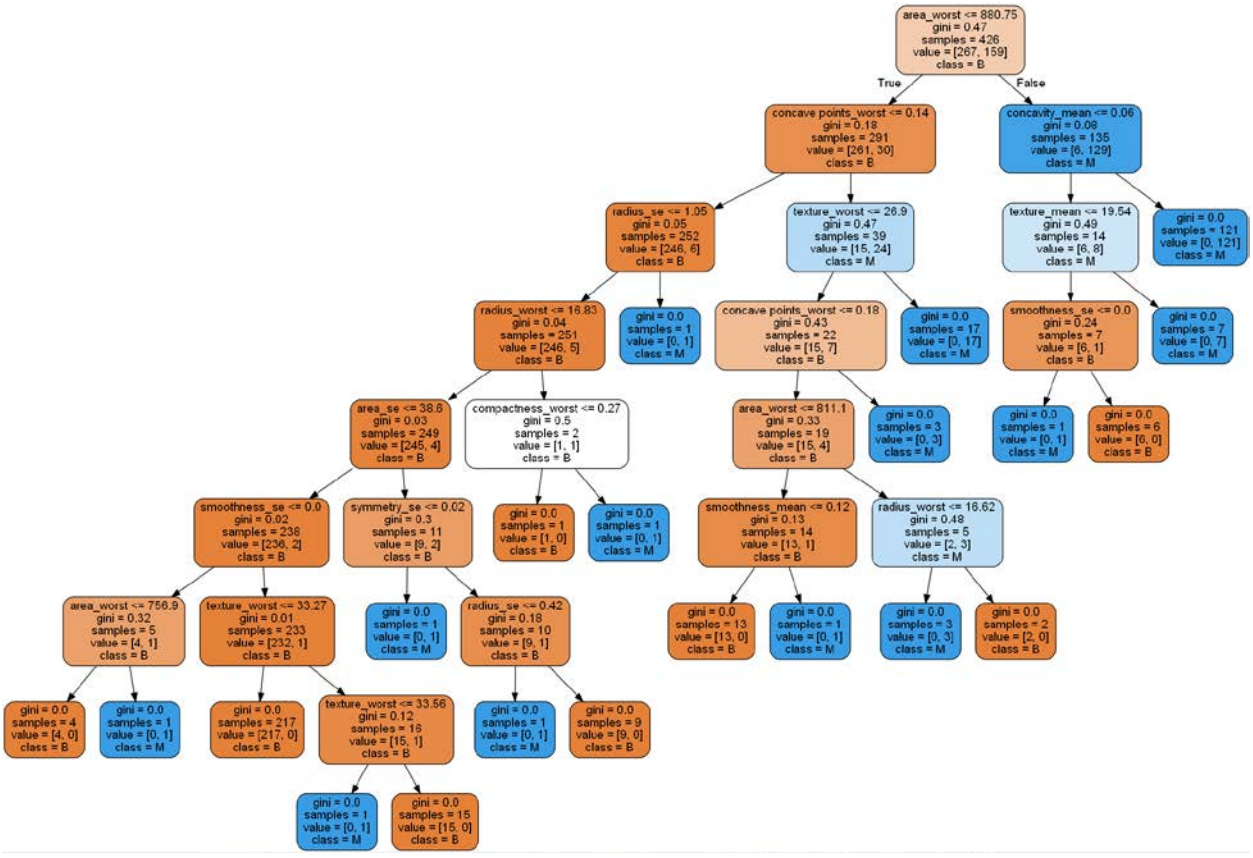
Decision tree is one of the most used ML classifiers. Decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers the to the question; and the leaves represent the actual output or class label.

When defining Decision Trees, one should first define criterion by which to measure the quality of a split. In python supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. In my analysis I chose default criterion i.e. 'gini'.

Gini Index for Binary Target variable is

$$Gini\ index = 1 - P^2(Target=0) - P^2(Target=1) = 1 - \sum_{t=0}^{t=1} P_t^2$$

From the decision tree below you can see gini index at each level, also *Value*- number of observations per each group and *class* that shows to which class observations satisfying previous node criteria now belong. I didn't define max depth, so it goes to the maximum depth possible. At this point one can decide at which point i.e. depth to cut tree.I think I would define depth to be 5 since I can see that at that depth predictions are good. As you can see from the tree, at gini=0.5 i.e. when there is a tie, color of the node is white, but when one class outnumbers other one, then node takes corresponding color. In our case orange for Benign and blue for Malignant. The more intense color, the lower is gini index i.e. more accurate is node prediction.

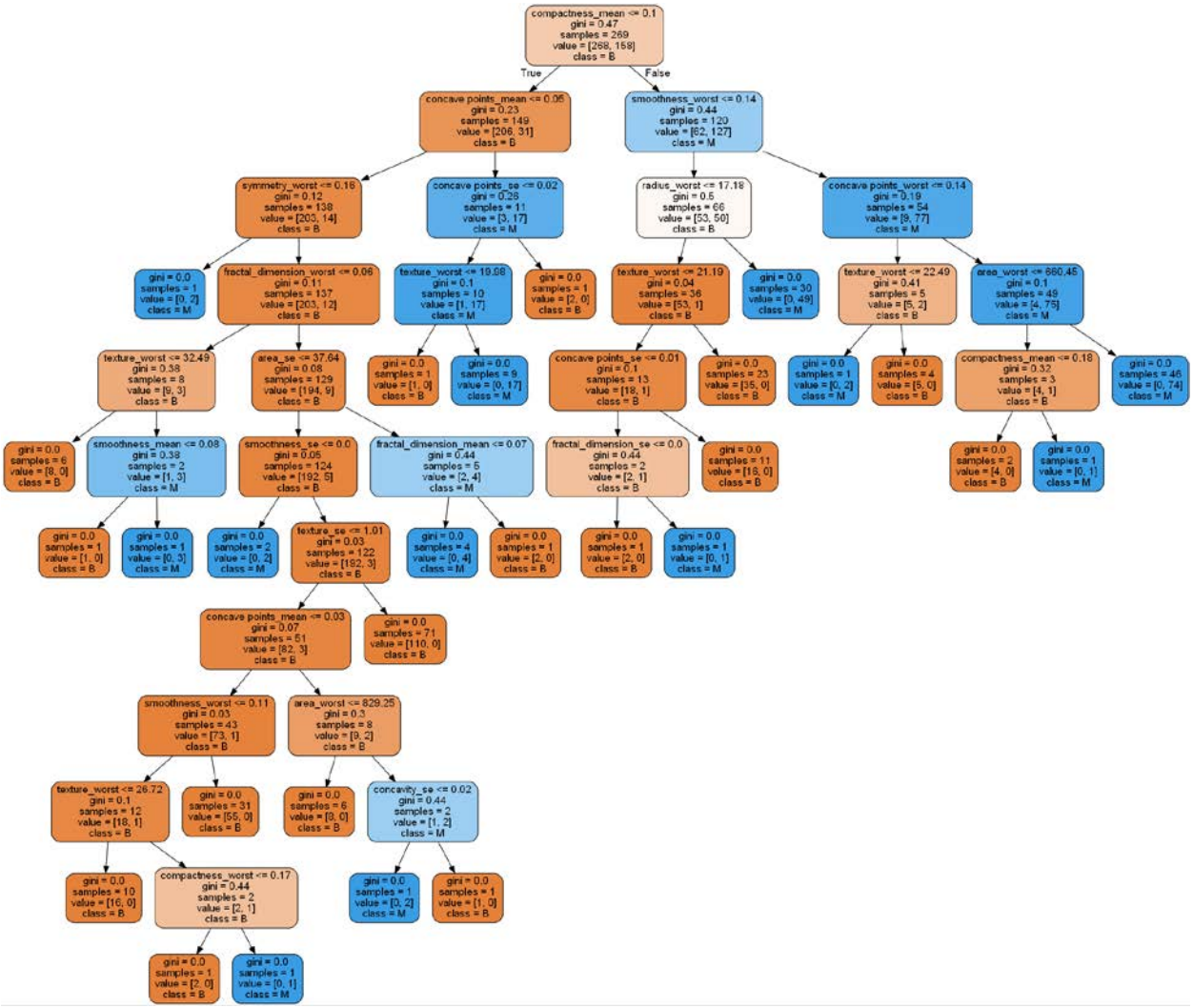


As for the previous algorithms, I calculated training accuracy which is ***100%***-overfitting. CV testing scores are 98%, 90%, 90%, 93% making average of ***93.1%.***

5. Random Forest Classifier

As we know Random forest classifier creates a set of decision trees from randomly selected subset of training set. It then aggregates the votes from different decision trees to decide the final class of the test object. So, most of the things are similar to Decision Tree but now we have bunch of trees. For my analysis I defined number of trees to be 100, for criterion I still use gini index and don't define maximum

depth. In the figure below you can see Decision tree i.e. 1 of 100 that are present in the Random Forest Classifier.



I calculated training accuracy which is ***100%***- overfitting too. CV testing scores are 100%, 92%, 90%, 98% making average of ***95.2%.*** i.e. Random Forest becomes one of the best.

Extra approach

In a search of the best results, I tried running Logistic regression with stepwise selection on SAS, since there is no such option in python, with p-entry= 0.05 and p-leave=0.051. I found these predictors to be significant: radius_se, area_worst, concavity_worst, compactness_se, texture_worst and concave points_worst. I ran all previous algorithms using just these predictors and came up with the results that weren't that different from when I used all 30 predictors. However, we will always prefer algorithm that uses less number of predictors. In the table below you can see the results for all algorithms using all predictors or just 6 that I just mentioned.

Name of Algorithm	# of input variables	Training Accuracy	Average of 4-fold
Decision Tree Classifier	6	100.00	94.54
Decision Tree Classifier	30	100.00	92.70
Random Forest Classifier	6	100.00	95.95
Random Forest Classifier	30	100.00	95.07
Logistic Regression-Lasso	6	96.60	95.42
Logistic Regression-Lasso	30	95.78	94.89
Logistic Regression-Ridge	6	94.90	92.78
Logistic Regression-Ridge	30	95.95	94.72
SVM (kernel=gausian)	6	99.47	65.55
SVM (kernel=gausian)	30	100.00	62.74
SVM (kernel=linear)	6	95.78	95.42
SVM (kernel=linear)	30	96.66	94.72
SVM (kernel=sigmoid)	6	62.74	62.74
SVM (kernel=sigmoid)	30	62.74	62.74
Kneighbor Classifier (distance=Manhattan)	6	94.37	92.26
Kneighbor Classifier (distance=Manhattan)	30	95.43	93.49
Kneighbor Classifier (distance=Euclidean)	6	93.49	92.09
Kneighbor Classifier (distance=Euclidean)	30	94.72	92.97
Kneighbor Classifier (distance=Minkowski p=3)	6	93.32	92.26
Kneighbor Classifier (distance=Minkowski p=3)	30	94.55	93.14

Conclusion:

After running 5 most popular Supervised learning algorithms I can conclude that for Breast Cancer dataset Random Forest Classifier, SVM with linear kernel and Logistic Regression with Lasso penalty give the highest accuracy results. I later figured out that running algorithm just for 6 most significant gave for some algorithm even better results and for other the difference was small. For my predictions I would use Top 3 algorithms using both: all predictors and only 6 best predictors, and then compare results.