Breast Cancer Classifiers and Comparison

by
Christian Hill

1. INTRODUCTION (The problem/task I am solving)

This paper covers the analysis and comparison of different Machine Learning classifiers preformed on the UCI Cancer Dataset from the University of Wisconsin Hospital, Madison from DR. William H. Wolberg. For the project my goal is to correctly predict benign and malignant tumors using different classifiers (SVM, KNN, Decision Trees, and MLP), then compare the different classifiers to determine which one is best for the dataset as well as detecting cancer.

2. THE DATA SET

In my dataset there are 569 instances with 30 attributes including the class label. My dataset was already completely preprocessed by the scikit-learn library and properly documented, but some of the data needed to be scaled for some of the classifiers. I also needed to create the training, test, subsets for the data before training.

3. FEATURES

While I had planned to try some feature engineering or feature selection for this project I didn't have time to properly execute and it wasn't necessary for my classifiers. Because this is a somewhat famous dataset, the features were very high quality, preprocessed, as well as documented.

From print(cancer.DESCR):

Attribute Information:

- radius (mean of distances from center to points on the perimeter)

- texture (standard deviation of gray-scale values)

- perimeter

- area

- smoothness (local variation in radius lengths)

- compactness (perimeter^2 / area - 1.0)

- concavity (severity of concave portions of the contour)

- concave points (number of concave portions of the contour)

- symmetry

- 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.

- class:

  - WDBC-Malignant

  - WDBC-Benign

:Summary Statistics:

	Min	Max
radius (mean):	6.981	28.11
texture (mean):	9.71	39.28
perimeter (mean):	43.79	188.5
area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04

texture (worst): 12.02 49.54

perimeter (worst): 50.41 251.2

area (worst): 185.2 4254.0

smoothness (worst): 0.071 0.223

compactness (worst): 0.027 1.058

concavity (worst): 0.0 1.252

concave points (worst): 0.0 0.291

symmetry (worst): 0.156 0.664

fractal dimension (worst): 0.055 0.208

========== ======

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

4. ALGORITHMS

The first algorithm I tested was KNN (K-Nearest Neighbor), I started with this algorithm because I wanted to visualize my data on a easier classifier to get a basis before implementing the harder algorithms. I also wanted to try and pick N to optimize the classifier. Next I used a SVM because I wanted to experience data scaling, tuning hyper-parameters like C or gamma, and choosing an appropriate kernel. I also chose an SVM because they're versatile and will perform well on this relatively small dataset. I then used a Decision Tree for almost the opposite reason as the SVM, with DT there is no need to pre-process, scale, or standardize the features before training, and the visualization opportunities were attractive as well. Finally, I used an MLP (Multi-layered Perceptron) mainly for the experience as well as the ability to tune the parameters.

5. Results

The two classifier that preformed the best were MLP and the SVM, with SVM having a higher accuracy on the test subset by .013. I'm not surprised that these two outperformed the others, mainly because I tuned hyperparameters of the two more than the decision tree and KNN. MLP and SVM also preformed very well when predicting the probabilities for the samples, with the DT coming in third and the KNN classifier coming in last.

KNN:

```
Accuracy of KNN n = 3,  on the training set: 0.953
Accuracy of KNN n = 3,  on the test set: 0.923

Accuracy of KNN n = 6,  on the training set: 0.941
Accuracy of KNN n = 6,  on the test set: 0.937
```

SVM:

```
accuracy on the training subset: 0.993
accuracy on the test subset: 0.972
```

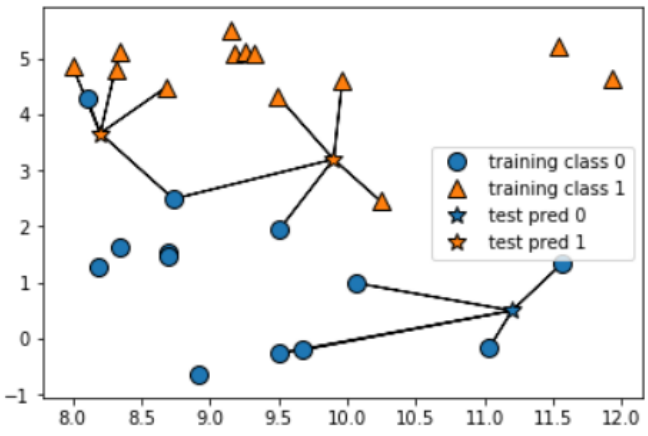
DT:

```
accuracy on the training subset: 0.991
accuracy on the test subset: 0.937
```

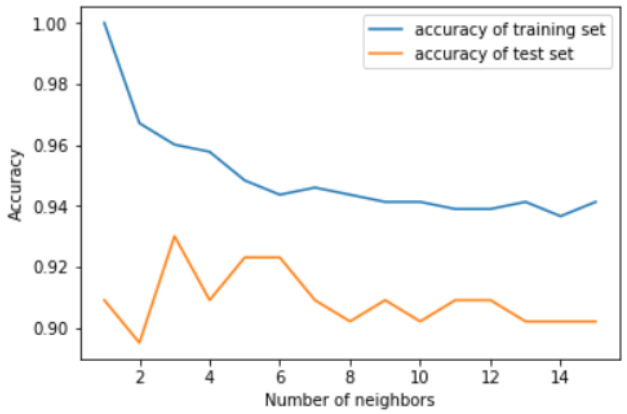
MLP:

```
accuracy on the training subset: 0.991
accuracy on the test subset: 0.958
```

Here's is some of the visualization done for KNN:
Stars are the random samples, blue circle represent malignant and triangle represent benign, and I preformed this mainly because this is a smaller data set with fewer features:

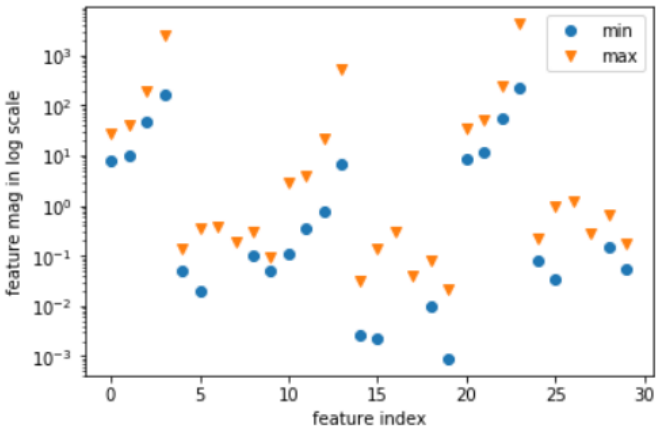


The next image is the where I found the best N to optimize the classifier:



The next images are from the SVM classifier:

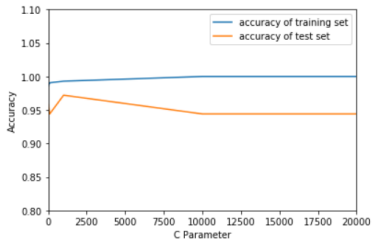
This image was taken because I was currently over-fitting on the training set as well as poor performance on the test, used this plot to decide that I needed to fix the scaling of the cancer data:



This image is where I tried different values of C in SVC to find the best value for the parameter:

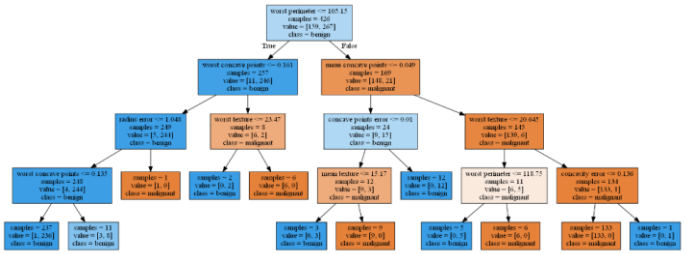
```
accuracy on the training subset: 0.986
accuracy on the test subset: 0.958

accuracy on the training subset: 0.991
accuracy on the test subset: 0.944

accuracy on the training subset: 0.993
accuracy on the test subset: 0.972

accuracy on the training subset: 1.000
accuracy on the test subset: 0.944

accuracy on the training subset: 1.000
accuracy on the test subset: 0.944

Out[244]: <matplotlib.legend.Legend at 0x26c16d5a6a0>
```

click to scroll output; double click to hide



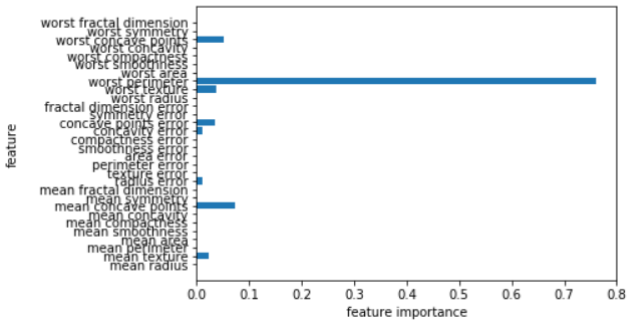
The next images are from the decision tree classifier:

This is the tree after applying pre-pruning to limit the number of decisions and avoid overfitting:



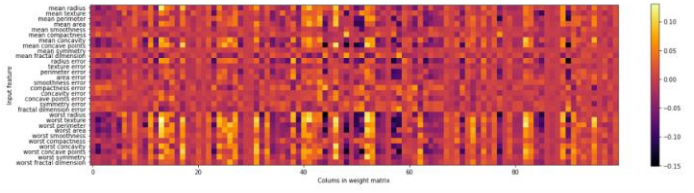
I then wanted to see why the tree was assembled the way it was so I viewed the feature importance:

```
Feature importance: [ 0.          0.02332154  0.          0.          0
  0.07269903  0.          0.          0.00999317  0.          0.
  0.          0.          0.01028778  0.03498231  0.          0.
  0.03660875  0.76144817  0.          0.          0.          0.
  0.05065925  0.          0.        ]
```



And finally this is my color bar from the MLP:

The columns are the number of hidden units, darker with more positive values and yellow with more negative. features with smaller weights in the hidden unit might be less important, this graph shows that the errors as well as mean smoothness, mean symmetry, as well as others don't have a major impact on the MLP decision making:



The next images are from the decision tree classifier: