Springboard Intensive Data Science Course
Breast Cancer Classification Project
Final Report

12 April 2018

Mentor: Pavitraa Parthasarathy

Student: Jing-Rui Li

1.Introduction - Background

Breast cancer is one of the most dangerous form of cancers in the world. Annually, Breast caner kills an estimated 20,000 people in the US. If breast cancer is recognized and treated early, it is almost curable, with a 80 to 90% 5-year survival rate when treated with simple excision. Early diagnosis of the breast cancer will help save thousands of lives.

The Breast Cancer project aims to develop a system that takes breast features as inputs, and outputs the likelihood that the patient is malignant, in order to facilitate early diagnosis of breast cancer. Unique and relevant features such as statistical measures (mean, standard deviation and covariance) of the breast's shape information (area, diameter, compactness and asymmetry) was extracted from the pre-processed images. Texture information, obtained from a gray level covariance matrix was also included. A trained classifier will then compute the likelihood that the lesion is malignant and return this percentage to the user.

Link to the dataset:
http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

2. Dataset

This project uses the Breast Cancer Wisconsin (Diagnostic) Database to create a classifier that can help diagnose patients. 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. The dataset of the breast cancer information can be accessed from build-in python package sklearn.dataset.load_breast_cancer. (Link to the dataset:
http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html).

Data Set Characteristics:

:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class

: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

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

References

- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages 570-577, July-August 1995.
- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) 163-171.

3. Data Visualization and Pre-processing

Firstly, Let's take a look at the number of Benign and Maglinant cases from the dataset. From the output shown below, there are 357 cases are benign (value =1) while 212 cases are malignant(value =0).

```
print(df.groupby('diagnosis').size())
```

```
diagnosis
0    212 (malignant)
1    357 (benign)
```

Next, I visualized the data using density plots to get a sense of the data distribution(Kernel density estimation). The kernel density estimate may be less familiar, but it can be a useful tool for plotting the shape of a distribution. Like the histogram, the KDE plots encodes the density of observations on one axis with height along the other axis.

.

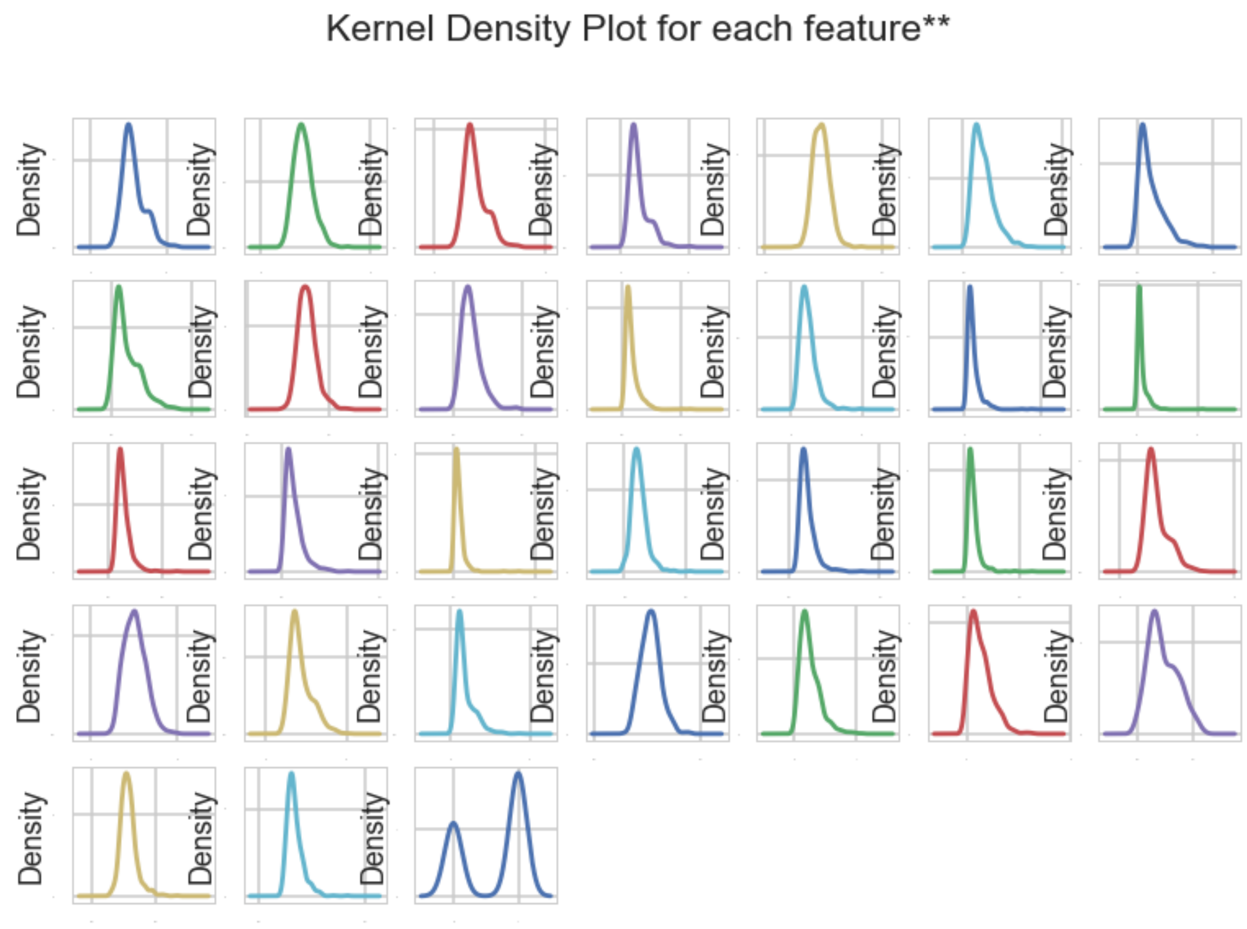


Figure 1. Kernel Density plot for each features in the dataset**

**The density plots above are for features as follows : 'mean radius' ,'mean texture' ,'mean perimeter' ,'mean area','mean smoothness' ,'mean compactness', 'mean concavity','mean concave points' ,'mean symmetry' ,'mean fractal dimension','radius error' 'texture error' 'perimeter error' ,'area error','smoothness error' ,'compactness error' 'concavity error','concave points error', 'symmetry error', 'fractal dimension error','worst radius' 'worst texture', 'worst perimeter' 'worst area','worst smoothness' 'worst compactness' 'worst concavity','worst concave points' ,'worst symmetry' ,'worst fractal dimension',’target(diagnosis)’.

From the outputs above, we can see the each feature of the dataset has a general Gaussian distribution (except for the target).

Then, It is good to check the correlations between the attributes.

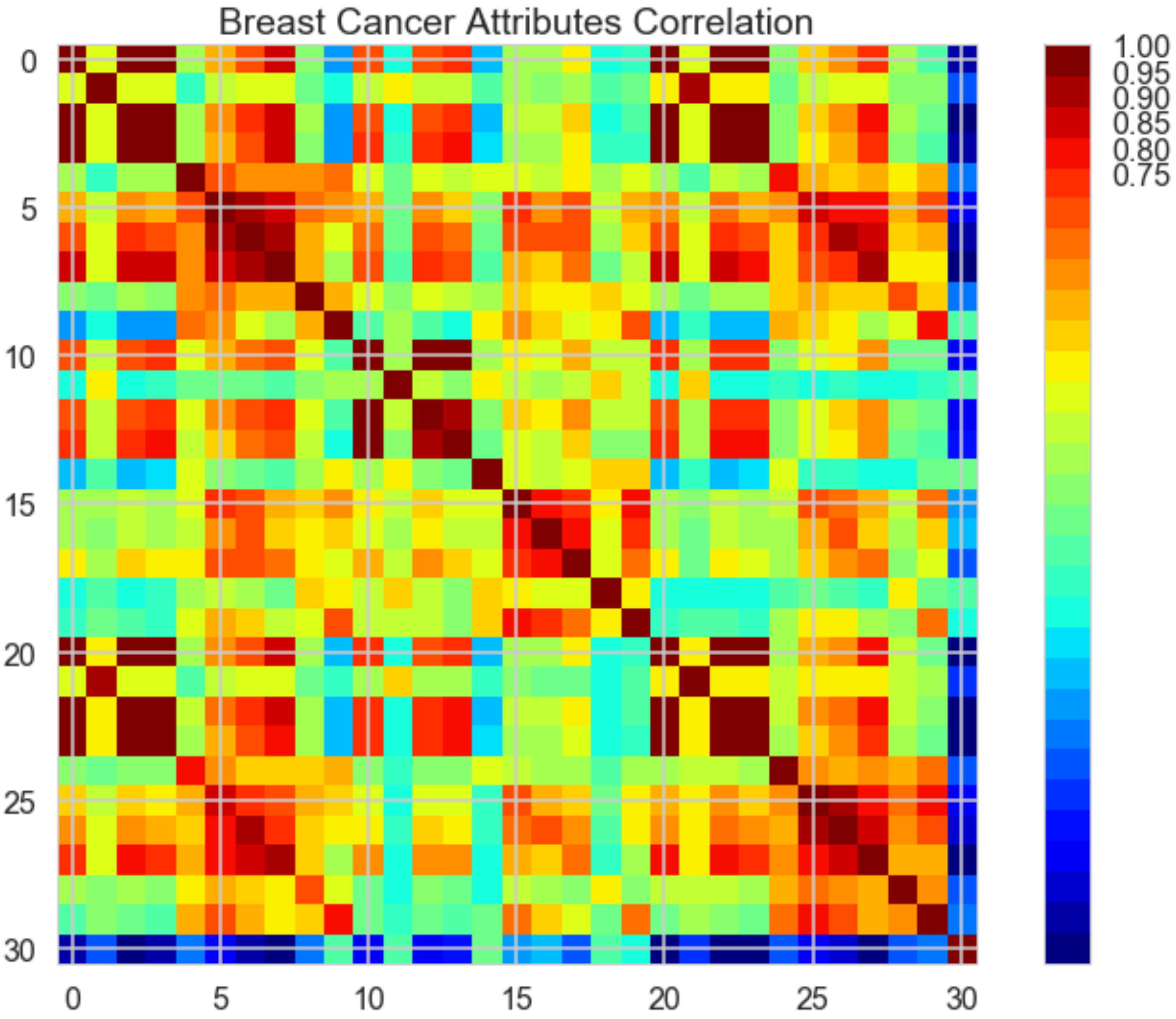


Figure 2 Correlation between each feature in the dataset

From the output graph below, The red around the diagonal suggests that attributes are correlated with each other. The yellow and green patches suggest some moderate correlation and the blue boxes show negative correlations.

3. Baseline Algorithm Checking with cross validation

From the dataset, we will analysis and build a model to predict if a given set of symptoms lead to breast cancer. This is a binary classification problem, and a few algorithms are appropriate for use. Since we do not know which one will perform the best at the point, we will do a quick test on the few appropriate algorithms with default setting to get an early indication of how each of them perform. We will use 5 and 10 fold cross validation for each testing.

The following non-linear algorithms will be used, namely: Classification and Regression Trees (CART), Linear Support Vector Machines (SVM), Gaussian Naive Bayes (NB) and k-Nearest Neighbors (KNN).

- **Result for 5 fold cross validation**

```
CART: 0.903557 (0.063812) (run time: 0.029673)
SVM: 0.657708 (0.070245) (run time: 0.039123)
NB: 0.921344 (0.050438) (run time: 0.012046)
KNN: 0.886166 (0.070369) (run time: 0.015470)
```

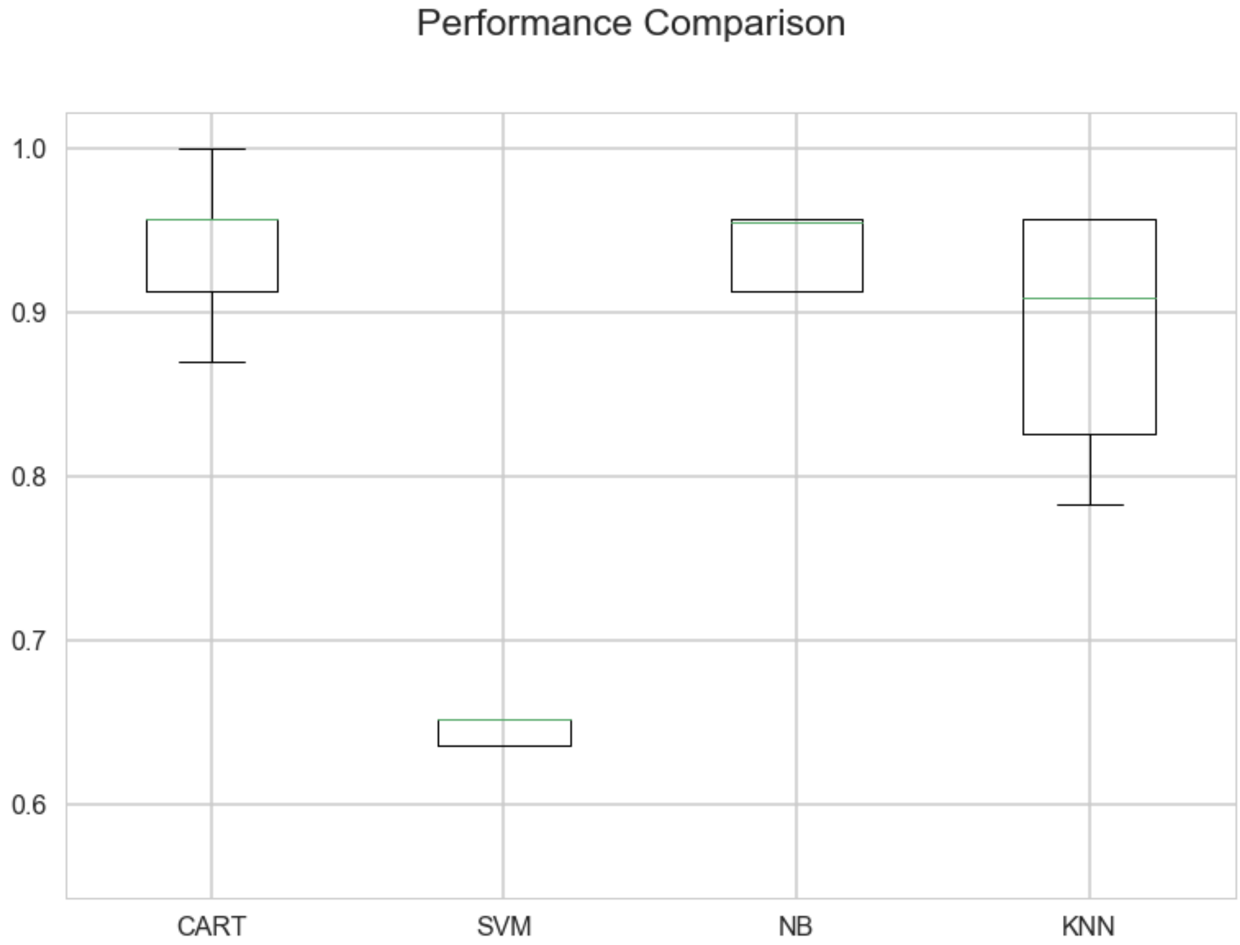


Figure 3. Five-Fold Cross validation comparison between CART, SVM, NB and KNN algorithms

- **Result for 10 fold cross validation**

```
CART: 0.913636 (0.094208) (run time: 0.026272)
SVM: 0.655303 (0.108957) (run time: 0.034203)
NB: 0.930303 (0.073434) (run time: 0.011217)
KNN: 0.894697 (0.085405) (run time: 0.021682)
```

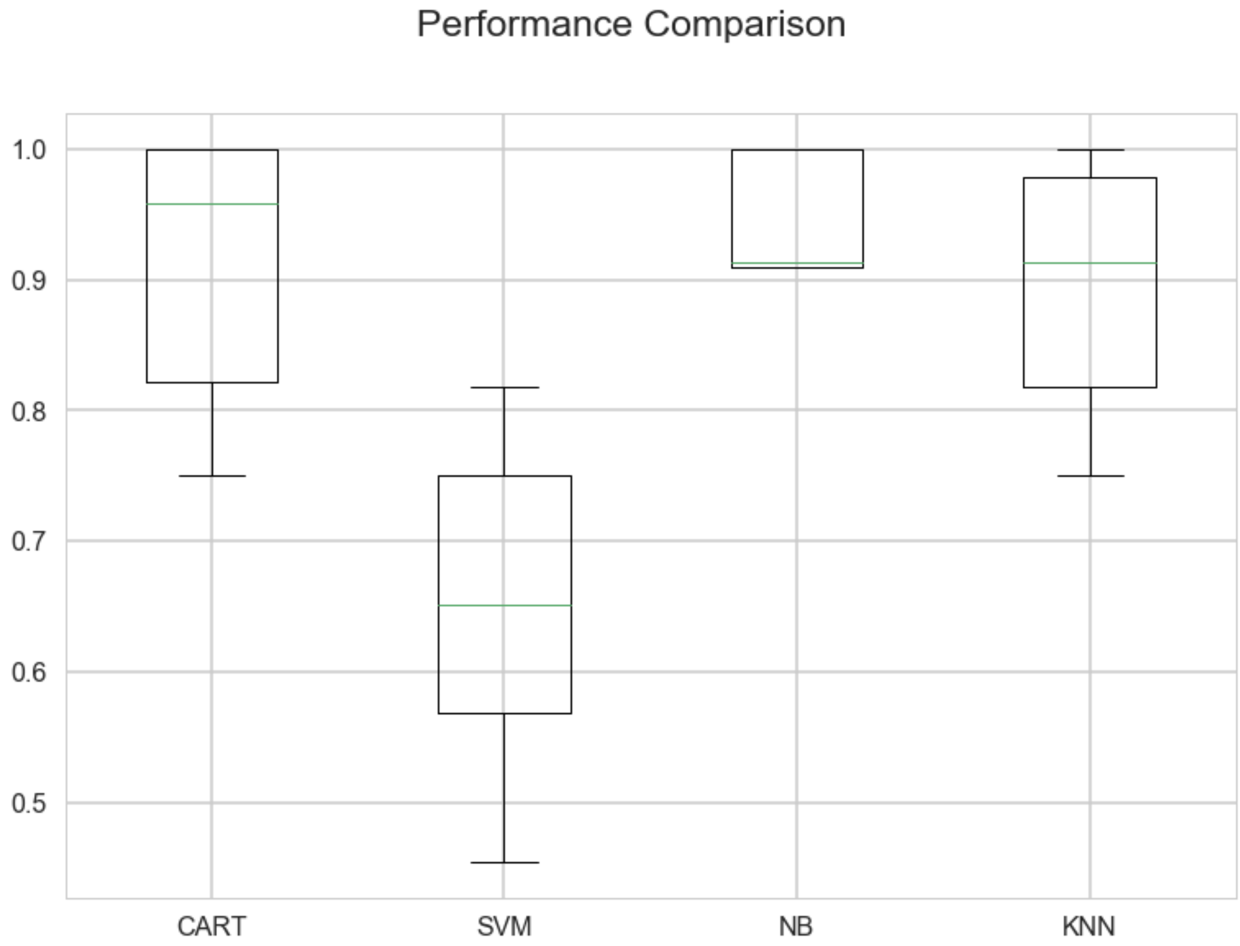


Figure 4. Ten-Fold Cross validation comparison between CART, SVM, NB and KNN algorithms

From the initial run, it looks like NB, KNN and CART performed the best given the dataset (all above 89% mean accuracy). Support Vector Machine has a surprisingly bad performance here. However, if we standardize the input dataset, it's performance should improve.

The performance of the few machine learning algorithm could be improved if a standardized dataset is being used. The improvement is likely for all the models. I will use pipelines that standardize the data and build the model for each fold in the cross-validation test harness. That way we can get a fair estimation of how each model with standardized data might perform on unseen data.

```
ScaledCART: 0.914396 (0.037331) (run time: 0.113696)
ScaledSVM: 0.964879 (0.038621) (run time: 0.074927)
ScaledNB: 0.931932 (0.038625) (run time: 0.033946)
ScaledKNN: 0.958357 (0.038595) (run time: 0.053220)
```

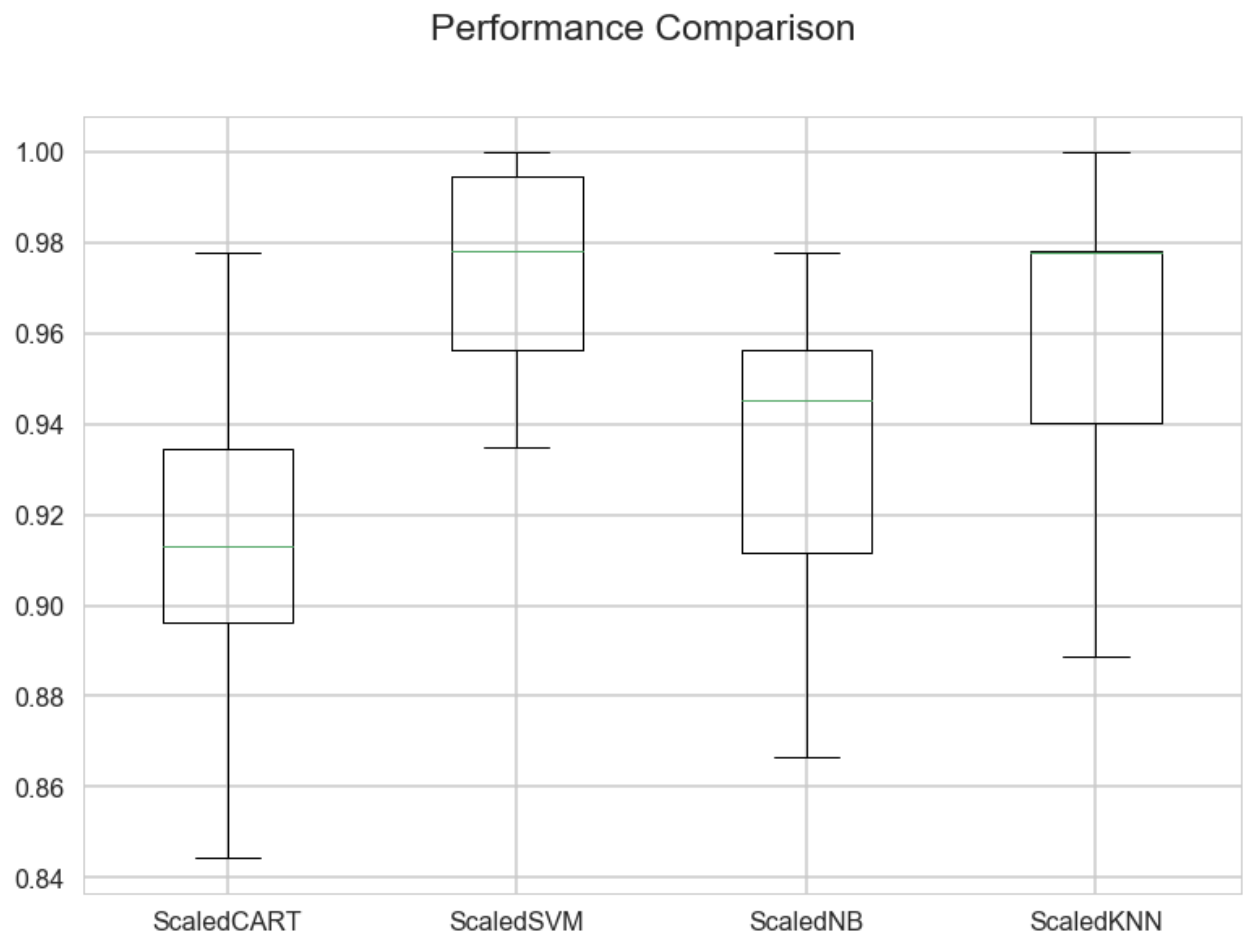


Figure 5. Ten-Fold Cross validation comparison between CART, SVM, NB and KNN algorithms after standardization

From the performance figure above, we can see the SVM algorithm has a great improvement with the standardized data.

Thus, I decide to focus on SVM for the algorithm tuning. We can tune two key parameter of the SVM algorithm - the value of C and the type of kernel. The default C for SVM is 1.0 and the kernel is Radial Basis Function (RBF). We will use the grid search method using 10-fold cross-validation with a standardized copy of the sample training dataset. We will try over a combination of C values and the following kernel types 'linear', 'poly', 'rbf' and 'sigmoid'.

```
Best: 0.969231 using {'C': 2.0, 'kernel': 'rbf'}
0.964835 (0.026196) with: {'C': 0.1, 'kernel': 'linear'}
0.826374 (0.058723) with: {'C': 0.1, 'kernel': 'poly'}
0.940659 (0.038201) with: {'C': 0.1, 'kernel': 'rbf'}
0.949451 (0.032769) with: {'C': 0.1, 'kernel': 'sigmoid'}
0.962637 (0.029474) with: {'C': 0.3, 'kernel': 'linear'}
0.868132 (0.051148) with: {'C': 0.3, 'kernel': 'poly'}
0.958242 (0.031970) with: {'C': 0.3, 'kernel': 'rbf'}
0.953846 (0.043579) with: {'C': 0.3, 'kernel': 'sigmoid'}
0.956044 (0.030933) with: {'C': 0.5, 'kernel': 'linear'}
```

```
0.881319 (0.050677) with: {'C': 0.5, 'kernel': 'poly'}
0.964835 (0.029906) with: {'C': 0.5, 'kernel': 'rbf'}
0.953846 (0.026785) with: {'C': 0.5, 'kernel': 'sigmoid'}
0.953846 (0.031587) with: {'C': 0.7, 'kernel': 'linear'}
0.885714 (0.038199) with: {'C': 0.7, 'kernel': 'poly'}
0.967033 (0.037271) with: {'C': 0.7, 'kernel': 'rbf'}
0.953846 (0.028513) with: {'C': 0.7, 'kernel': 'sigmoid'}
0.951648 (0.028834) with: {'C': 0.9, 'kernel': 'linear'}
0.887912 (0.038950) with: {'C': 0.9, 'kernel': 'poly'}
0.967033 (0.037271) with: {'C': 0.9, 'kernel': 'rbf'}
0.958242 (0.033272) with: {'C': 0.9, 'kernel': 'sigmoid'}
0.953846 (0.026546) with: {'C': 1.0, 'kernel': 'linear'}
0.890110 (0.038311) with: {'C': 1.0, 'kernel': 'poly'}
0.967033 (0.033027) with: {'C': 1.0, 'kernel': 'rbf'}
0.964835 (0.031496) with: {'C': 1.0, 'kernel': 'sigmoid'}
0.956044 (0.025765) with: {'C': 1.3, 'kernel': 'linear'}
0.894505 (0.039427) with: {'C': 1.3, 'kernel': 'poly'}
0.967033 (0.028188) with: {'C': 1.3, 'kernel': 'rbf'}
0.958242 (0.033224) with: {'C': 1.3, 'kernel': 'sigmoid'}
0.958242 (0.024765) with: {'C': 1.5, 'kernel': 'linear'}
0.896703 (0.039791) with: {'C': 1.5, 'kernel': 'poly'}
0.967033 (0.028188) with: {'C': 1.5, 'kernel': 'rbf'}
0.945055 (0.034215) with: {'C': 1.5, 'kernel': 'sigmoid'}
0.956044 (0.021766) with: {'C': 1.7, 'kernel': 'linear'}
0.903297 (0.033409) with: {'C': 1.7, 'kernel': 'poly'}
0.967033 (0.024479) with: {'C': 1.7, 'kernel': 'rbf'}
0.956044 (0.035354) with: {'C': 1.7, 'kernel': 'sigmoid'}
0.956044 (0.021766) with: {'C': 2.0, 'kernel': 'linear'}
0.909890 (0.033680) with: {'C': 2.0, 'kernel': 'poly'}
0.969231 (0.022370) with: {'C': 2.0, 'kernel': 'rbf'}
0.947253 (0.028116) with: {'C': 2.0, 'kernel': 'sigmoid'}
```

For the training datasest, we can see the most accurate configuration was SVM with an RBF kernel and C=2.0.

4. Applying various forecasting models

4.1 SVM Analysis on Test Dataset

From the baseline processing, I selected SVM out of four algorithms as the most accurate model for forecasting the breast cancer data. Using the SVM tuning method, I further selected the parameters with RBF kernel and C=2.0. Now, I would like to use the selected parameters to estimate the accuracy of the test dataset.

SVM Test Dataset Result

- **Accuracy score**

```
print("Accuracy score %f" % accuracy_score(Y_test, predictions))
print(classification_report(Y_test, predictions))
```

```
Accuracy score 0.657895
                  precision    recall  f1-score   support
```

```
          0       0.00      0.00      0.00        39
          1       0.66      1.00      0.79        75

avg / total       0.43      0.66      0.52       114

avg / total       0.99      0.99      0.99       114
```

- **Confusion Matrix**

```
print(confusion_matrix(Y_test, predictions))
[[39  0]
 [ 1 74]]
```

From the confusion matrix, there is only 1 case of mis-classification. The performance of this algorithm is expected to be high given the symptoms for breast cancer should exhibit certain clear patterns.

4.2 Naïve Bayes Analysis on Test Dataset

- **Accuracy score**

```
print("Accuracy score %f" % accuracy_score(Y_test, predictions))
print(classification_report(Y_test, predictions))
Accuracy score 0.657895
             precision    recall  f1-score   support

          0       0.00      0.00      0.00        39
          1       0.66      1.00      0.79        75

avg / total       0.43      0.66      0.52       114
```

- **Confusion Matrix**

```
[[33  6]
 [ 3 72]]
```

From the confusion matrix, we can see that there are 3 cases of mis-classification for Gaussian Naïve Bay es algorithm.

4.2 Random Forest Analysis

Although we found that SVM is one of the best model for the dataset, I would like to further explore the dataset and rank the importance of each feature using another algorithm-random forest analysis. Likewise, the data was split into test and train datasets for the testing.

```
importance
feature
mean concave points          0.305
mean concavity               0.228
mean area                    0.135
mean perimeter               0.109
mean texture                 0.057
mean radius                  0.056
mean compactness             0.051
mean smoothness            0.026
mean symmetry              0.020
mean fractal dimension     0.013
```

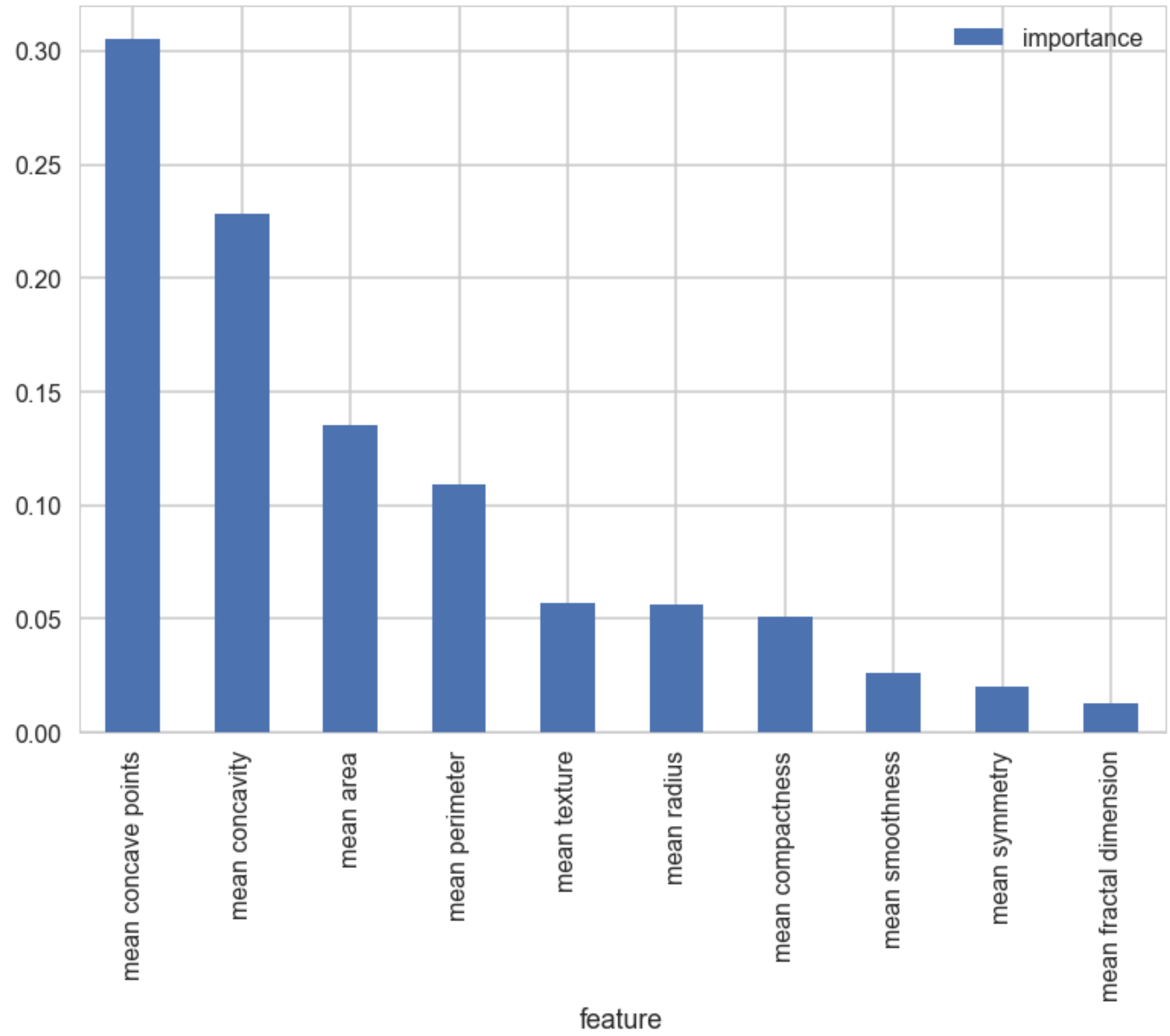
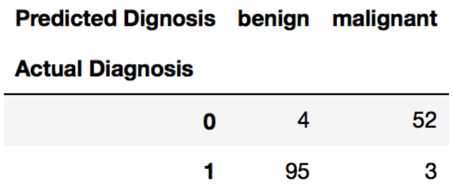


Figure 6. Importance ranking for the top features within the dataset

Secondly, I did confusion matrix analysis. The columns are the diagnosis we predicted for the test data and the rows are the actual result for the test data.



Predicted Dignosis	benign	malignant
Actual Diagnosis		
0	4	52
1	95	3

** 1 represents benign and 0 represents malignant. From the confusion matrix result, 4 out of 99 of benign diagnosis is predicted wrong (accuracy of 95.59%), while 3 out of 55 malignant result is predicted wrong (accuracy of 94.54%).

5. Thoughts and Conclusion

In this project, I compared prediction accuracy among four different algorithms. Through the comparison, I found that SVM did not perform well before standardize the data, however, it turned out to be the best estimation algorithm with cross validation score of 96.4% after standardization. It reminded me that when doing analysis on an unfamiliar dataset, it is critical to standardize the data before making any comparison or further analysis.

Later through the project, I did tuning SVM and select RBF and C=2.0 as the best classifiers with accuracy of 96.92%. The tuning process helped me to find the best parameters for the dataset, which help to improve the accuracy score in the test data. This reminds me the importance of selecting the appropriate classifiers before doing the prediction.

Finally, the feature ranking helps me get a better understanding of the importance of each feature. This will help improve the model in the further to make predictions when some features are missing(modify the weight of each feature).

Citation

https://pandas.pydata.org/pandas-docs/version/0.18.1/visualization.html

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://stackoverflow.com/questions/41458834/how-is-scikit-learn-cross-val-predict-accuracy-score-calculated

https://www.programcreek.com/python/example/91159/sklearn.model_selection.cross_val_predict