Assignment 2. Part 1 – Binary Classification with Logistic Regression and k-Nearest Neighbours

In part 1 of this assignment, logistic regression and k-nearest neighbours for binary classification has been experimented. The "Wisconsin breast cancer" data set was used for the experiments. There are 30 features and two classes in this dataset: "malignant" (the positive class), and "benign" (the negative class).

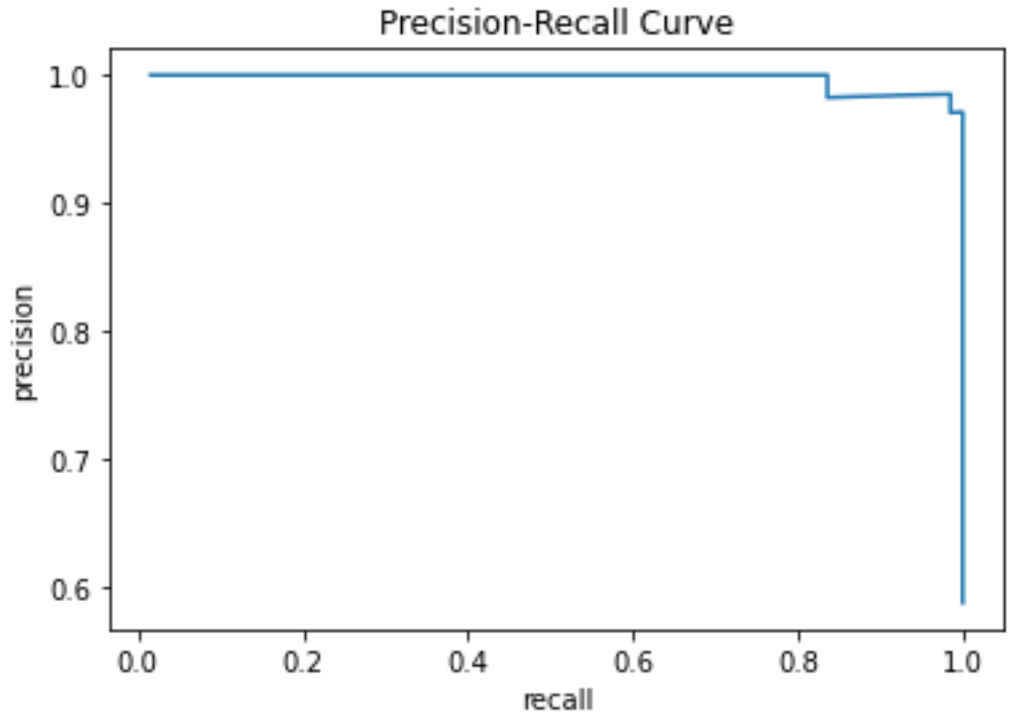
Self-Implementation of Logistic Regression:

Vectors of parameters:

[0.5531915 -0.57656993 -0.38620744 -0.53096624 -0.69846698 -0.48386937
0.70627354 -0.86399076 -1.12273601 -0.0129547 0.37105532 -1.4442237
-0.10747822 -0.86497436 -1.12432708 -0.09845602 0.95192182 -0.01081141
-0.4315786 0.29087216 0.8627654 -1.19440698 -1.47620015 -1.00412259
-1.21581294 -1.0059328 0.15757826 -1.12693381 -1.18097163 -0.85777862
-0.41049211]

The misclassification rate = 0.017543859649122806
The F1 score on the test data = 0.9850746268656716

Below shows the precision/recall (PR) curve that considers all possible thresholds.



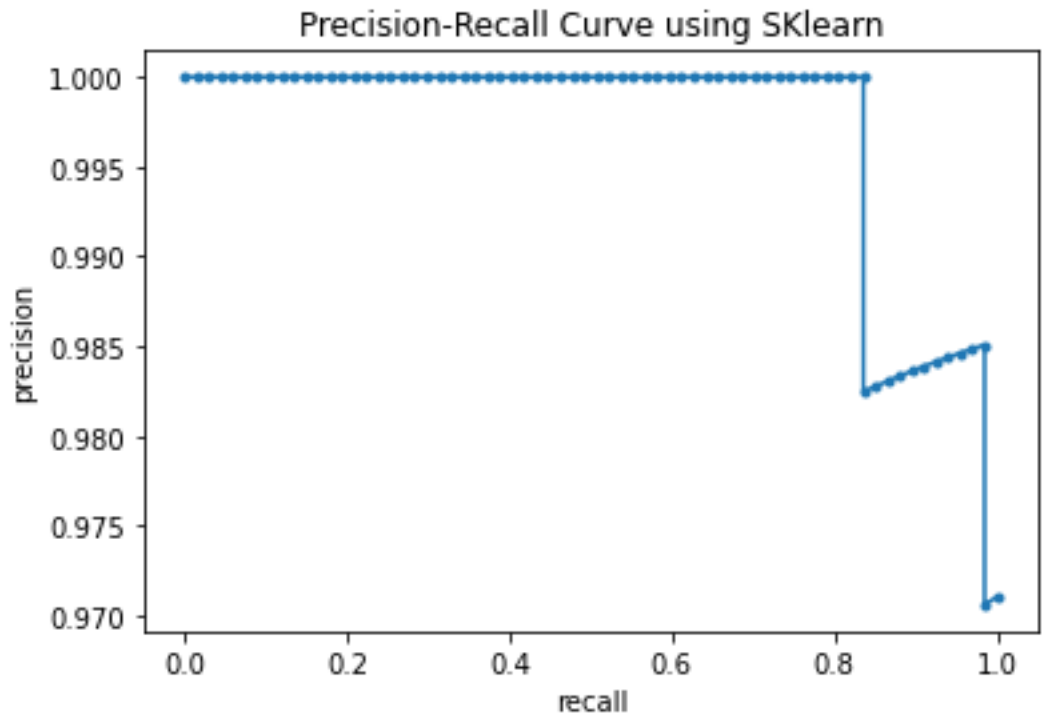
Scikit-Learn Implementation of Logistic Regression:

Vectors of parameters:

[-0.46732061 -0.30870548 -0.43600504 -0.52533515 -0.37320266 0.55425808
-0.69987562 -0.94444544 -0.060741 0.28852253 -1.1349591 -0.0199271
-0.6092074 -0.86998242 -0.08867453 0.63525478 0.08933266 -0.39648537
0.2121341 0.67547721 -0.95384134 -1.23397878 -0.79710496 -0.93338914
-0.70720501 0.12300704 -0.89646804 -0.90736645 -0.63634771 -0.36050551]

The misclassification rate = 0.01754385964912286
The F1 score on the test data = 0.9850746268656716

Below shows the precision/recall (PR) curve that considers all possible thresholds. *Note: scikit-learn utilities was used to plot the PR curve.*



Self-Implementation of k-Nearest Neighbour

k-Nearest Neighbour	Cross-validation errors for Kfold = 5	Average CV Error
k = 1	0.04395604395604396 0.07692307692307693 0.05494505494505494 0.04395604395604396 0.07692307692307693	0.05934065934065935
k = 2	0.04395604395604396 0.07692307692307693 0.05494505494505494 0.04395604395604396 0.07692307692307693	0.05934065934065935

k = 3	0.03296703296703297 0.03296703296703297 0.03296703296703297 0.01098901098901099 0.054945054945054944	0.032967032967032975
k = 4	0.02197802197802198 0.03296703296703297 0.03296703296703297 0.01098901098901099 0.054945054945054944	0.03076923076923077
k = 5	0.02197802197802198 0.054945054945054944 0.03296703296703297 0.02197802197802198 0.04395604395604396	0.035164835164835165

The best k is: 4
The test error is 0.02631578947368421
The F1 Score is: 0.9781021897810218

Scikit-Learn Implementation of k-Nearest Neighbour

k-Nearest Neighbour	Cross-validation errors for Kfold = 5	Average CV Error
k = 1	0.07692308 0.04395604 0.05494505 0.05494505 0.06593407	0.05934065934065935
k = 2	0.06593407 0.04395604 0.04395604 0.07692308 0.06593407	0.05934065934065935
k = 3	0.03296703 0.03296703 0.04395604 0.02197802 0.01098901	0.02857142857142847
k = 4	0.03296703 0.03296703 0.02197802 0. 0.01098901	0.01978021978021971
k = 5	0.03296703 0.04395604 0.02197802 0.01098901 0.01098901	0.02417582417582409

The best k calculated by sci-kit learn is: 4
The test error calculated by scikit-learn kNeighborsClassifier is: 0.04385964912280704
The F1 score calculated by scikit-learn is: 0.9624060150375939

Conclusion

	Self-Implementation of Logistic Regression	Scikit-Learn implementation of Logistic Regression	Self-Implementation of k-Nearest Neighbour	Scikit-Learn implementation of k-Nearest Neighbour
Test Errors	0.017543859649122806	0.01754385964912286	0.02631578947368421	0.04385964912280704
F1 score	0.9850746268656716	0.9850746268656716	0.9781021897810218	0.9624060150375939

The results of my implementation compared with the results obtained using scikit-learn is very close in numbers. The best model of the four models is tied between my own implementation of Logistic Regression and scikit-learn's model of Logistic Regression. We can also see that for k-Nearest Neighbour, the misclassification rate (test error) obtained from my own implementation is lower than the results obtained from scikit-learn. The F1 score of my own implementation is greater than scikit-learn as well.

Assignment 2. Part 2 – Ensemble Methods

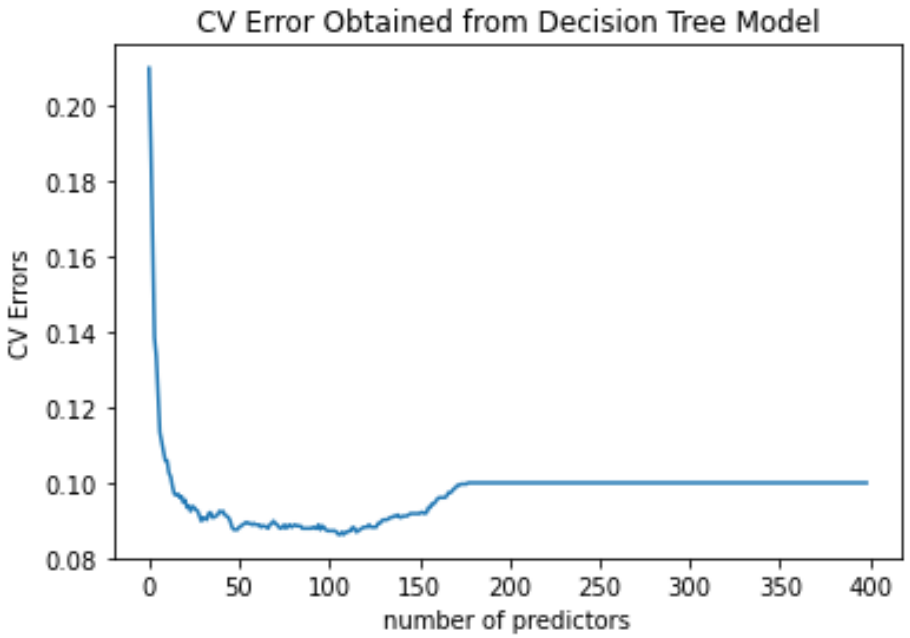
In part 2 of this assignment, ensemble methods (bagging, random forest and boosting) has been experimented. The "spambase" data set was used for the experiments. There are 57 features and two classes in this dataset: "spam" (label 1), and "nonspam" (label 0).

The classifiers to train are:

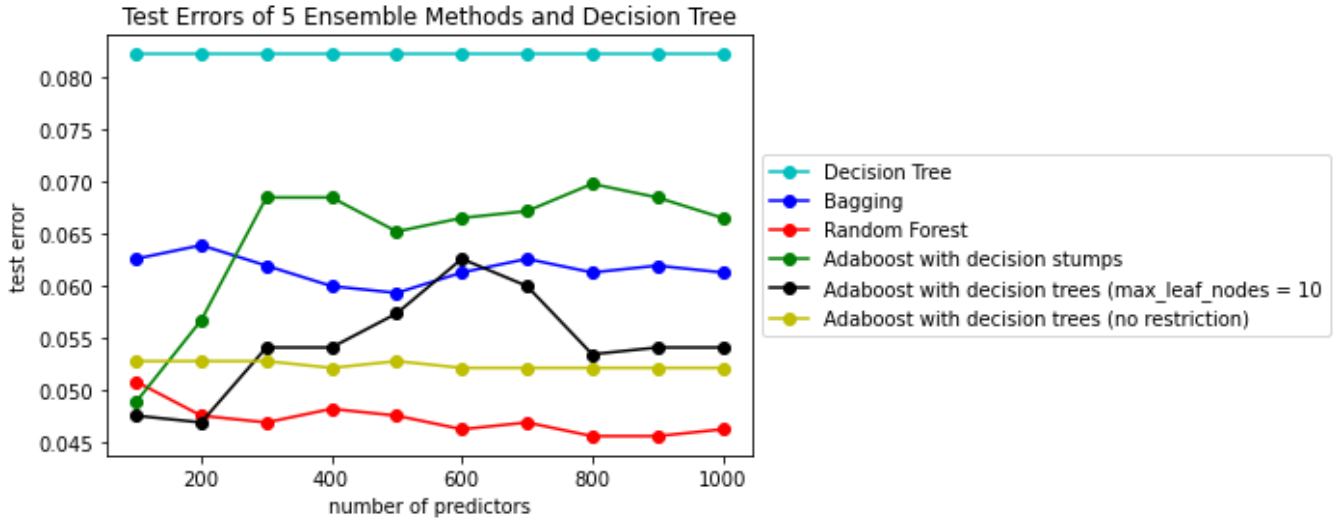
- One Decision Tree Classifier
- 50 Bagging Classifiers
- 50 Random Forest Classifiers
- 50 Adaboost Classifiers with decision stumps
- 50 Adaboost Classifiers with decision trees (max_leaf_nodes = 10)
- 50 Adaboost Classifiers with decision trees (no restriction)

Below shows the plot of the cross-validation obtained with the decision tree model for all values of the maximum number of leaves considered. It is shown that it decreases exponentially as number of predictors increases and reaches its lowest at around the 100s and starts to stay consistent at 10% as number of predictors increases.

The maximum number of leaves calculated is **107** with a test error of **0.08213820078226863.**



As training 50 classifiers for each of the five ensemble methods takes too long, only 10 classifiers were considered as number of predictors from integers from 100 to 1000 in increments of 100. The processor used to run this code is a 2.7 GHz Dual-Core Intel Core i5. The time it takes to run the code is around 10 minutes.



Bagging Classifier stays fairly consistent with the test errors all throughout the number of predictors with the lowest test error at 500 predictors.

Random Forest Classifier produces a low test error all throughout and gets even lower as number of the predictors increase.

The AdaBoost Classifier with decision stumps produces a fairly low test error for the first 200 predictors and continues to increase when the number of predictors get to around 300. The test error starts to stay around that range as number of predictors increases. It has one of the lower errors in the first 200 predictors but becomes the worst model (after Decision Tree classifier) as number of predictors increase. Decision stumps are known to be simple weak learners and are less prone to overfitting.

The AdaBoost Classifier with decision trees with at most 10 leaves as the base classifier has the lowest test error for the first 200 predictors. The test error increases as the number of predictors increases and it peaks at 600 predictors, being the third worst model at around 600 predictors. The test error decreases and stabilizes at around 800 predictors.

The AdaBoost Classifier with no restriction on the decision trees produces a consistent and fairly low test error as number of predictors increases.

As a result, for the first 200 predictors, the best model is the AdaBoost Classifier with decision trees at 10 leaves as the base classifiers (i.e. max_leaf_nodes = 10). However, as the number of predictors increase, the Random Forest Classifier produces the lowest test error. The Decision Tree Classifier has the highest error. Random Forest stays the lowest error all throughout and is the best model overall. The models all have fairly close errors and is only off by about ~1-2% from each other.