CAB320 Assignment 2

Alexander Santander: n9684026
Hrushikesh Lakkola: n9654780

Introduction

Machine learning has an ever broadening horizon in modern society with seemingly infinite applications.The task outlined in this report is essentially a small taste of this genre but was very worthwhile nonetheless. We were given a tabular set of data and told to run it through various well known classifiers, while being sure to model and shape the data to reap the most accurate results. The primary API used for this endeavor was “scikit-learn” due to its comprehensive coverage of what we needed. This report will methodically convey our implementation and results.

Methodology

Modelling the Dataset

To determine the best machine learning model to produce accurate outputs from the given dataset, four different classifiers were chosen: Naive Bayes, Decision Tree, Nearest Neighbours and Support vector machine. As classification (label prediction) within Machine learning tasks require outcome (Y) and predictor (X) measurements, the original dataset was split into two datasets in the ‘prepare_dataset’ function. The first dataset, otherwise known as the outcome/output, consisted of the tumor type labels corresponding to a ‘1’ for Malignant and ‘0’ for Benign. The second dataset consisted of 10 real-valued features for each cell nucleus.

The total dataset (X & Y) was then partitioned into training and testing datasets. For the classifiers utilizing hyperparameters (KNN, DT, SVM), the best hyperparameter value would be tuned using the training dataset. The chosen hyperparameter value would then be applied to the test set to determine the accuracy of the prediction.

Naive Bayes

The Naive Bayes classifier is based on Bayes’ theorem. It essentially predicts the outcome of a given set of data based on the data it has been provided. However, it can only do this through the assumption that every data point within a class is unrelated another within the same class. This assumption is binary in that it must be true and cannot be false and so there is no weighting or adjustment that can be made to improve the accuracy of the result. Therefore this classifier was not implemented with a hyperparameter.

Due to the lack of hyperparameter implementing this classifier was particularly straightforward. We simply fit the data and cross validated it a standard 10 times then returned the mean value. The GaussianNB() function did require us to convert our training data to floats, but this is simply a due to how the classifier was written.

Decision Tree (DT)

The Decision Tree classifiers goal is to predict the output of a target variable based on all the input variables it has been given. The basic structure of a decision tree involves a question, this question can be either a domain or boolean and is relevant to a class of the data it has been given. The interior node leads to the next question whereas the leaf node returns a probability of an event given all the decisions that have been made thus far. Given this understanding we decided to used the maximum depth as our hyperparameter for this classifier. This is because it will not only massively impact the output, but allow us to monitor the computation time and ensure not too many or too few iterations are performed.

Our optimal max_depth hyperparameter value search consisted of calling the 'DecisionTreeClassifier()' function in a for loop with a range of values. The range of values was chosen by trial and error and initially, a range of 1 to 50 was chosen. After looking at the resulting accuracies, we realized that the classifier was randomizing itself after depth values of 7-10 and accuracies after that range had insignificant variance. We then added a random seed to the random_state hyperparameter to display clearer results.

Nearest Neighbours (KNN)

The K-Nearest Neighbours classifier is widely considered to be a good entry point to machine learning due to its simplicity and effectiveness. It works by looking at all the data points it has been provided and categorizing each by analyzing what class the closest data points to it belong to. Depending upon the weight of "k" the data point in question will then be assigned to the appropriate class. The lower the value of k the greater the weight is placed upon its nearest neighbours. Thus it was a very simple decision to use "k" as our hyperparameter.

When implementing this classifier we first determined what range of k values would be appropriate for testing. It was found that the best "k" value was always below 15 and so for maximum efficiency only a range of 15 needs computation. However to better illustrate a trendline we decided upon a range of 50 due to how quickly the classifier computed. From here we simply iterated through this range and each time returning the mean of 10 cross validated values. We then re-entered the mean of all those values and were given our final cross validation score for our best "k".

Support Vector Machine (SVM)

Support Vector Machines seek to classify data through the use of vectors in a potentially highly dimensional space. These vectors separate the data with the help of a "soft margin" which essentially allows for outliers in order to find a much cleaner fit. However because we decided to use a Nonlinear Support Vector the linearity of the margin can be altered through the parameter "gamma". This parameter determines the weight of each data point relative to how close they are to the margin. The lower the value the more the margin will weave and snake around data points. This makes it very significant and so "gamma" was chosen as our hyperparameter.

The process used to find the best value for our hyperparameter and train our classifier is essentially the same as what was done in the KNN and DT sections, the only real hurdle being an appropriate range of values for "gamma". With some research we found that responds to extremely small numbers, after some educated guesses we decided upon 10^-6 to 10^-1 because it illustrates a clear and reasonable trendline.

Evaluation

Naive Bayes

Table 1: Errors and accuracies for GuassianNB classifier.

Type	Values
Cross-validated accuracy/scores	0.937
Training error	0.0585

Testing error	0.0526
Testing accuracy	0.947

The results showed a 93.7% cross validated score with a 94.7% testing accuracy. Due to the lack of any hyperparameters, only basic reporting was needed. As seen in table 1, it was surprising to see a 1% increase in accuracy from training to testing as it was expected that testing accuracy would be lower than training as the classifier had not seen the predictor (output values) in the training accuracy.

Decision Tree

Table 2: Errors and accuracies for Decision tree classifier.

Type	Values
Cross-validated accuracy/scores	0.929
Standard deviation of cross validated scores	0.00938
Training error	0.003906
Testing error	0.105
Testing accuracy	0.894

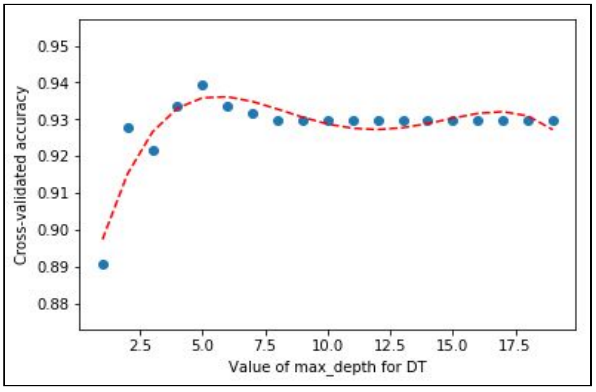

Figure 1: Cross-val training accuracies for range of max_depth values.

As seen in Figure 1 and table 2, the best cross-validated accuracy produced was 92.9% with a max_depth value of 5. Figure 1 also shows the accuracies plateauing after max_depth values of 5-10. Testing accuracy was at 89% accuracy with the testing dataset and showed a 3.9% drop from training accuracy although an error of 0.105 is acceptable.

K-Nearest Neighbour

Table 3: Errors and accuracies for K-Nearest Neighbours classifier.

Type	Values
Cross-validated accuracy/scores	0.937
Standard deviation of cross validated scores	0.00599
Training error	0.058
Testing error	0.0877
Testing accuracy	0.912

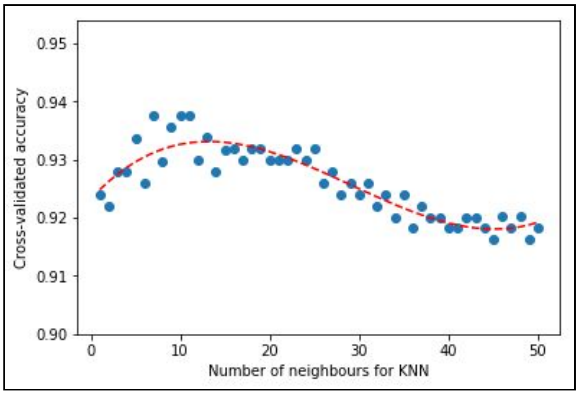

Figure 2: Cross-val training accuracies for range of max_depth values.
Figure 2 and table 3 show a cross-validation accuracy of 93.7% with the optimal amount of neighbours (k-value) being 7. Testing accuracy was 91.2% with a 2.5% drop in accuracy. A testing error of 0.0877 is within acceptable range. Figure 2 also shows the cross-validation accuracy for the training set decreasing after a peak around a k value of 10. This trend was seen in multiple executions of the program.

Support vector machine

Table 4: Errors and accuracies for SVM classifier.

Type	Values
Cross-validated accuracy/scores	0.935
Standard deviation of cross validated scores	0.123
Training error	0.0507
Testing error	0.035
Testing accuracy	0.964

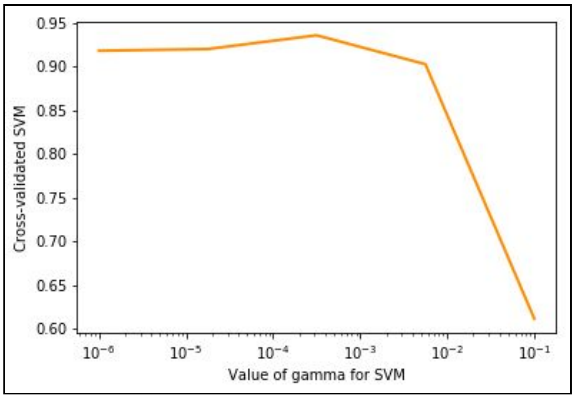

Figure 3: Cross-val training accuracies for range of gamma values.
Figure 3 and Table 2 show a cross-validation accuracy of 93.5% with the optimal gamma value being 0.000316 and a testing accuracy of 96.4%. The testing accuracy was 2.9% lower than the training accuracy. This was not expected as the training set would be given the output labels to train with and was expected to score a higher cross validation accuracy.

Conclusion

In conclusion, all four machine learning classifiers produced results on the testing dataset that were within acceptable accuracies. The most accurate classifier was the support vector machine classifier with a score of 96.4% on the testing dataset and the least accurate classifier was the decision tree classifier with a score of 89.4%.