CENG 499
Take Home Exam 3 Report

Ilter Taha Aktolga

January 8, 2021

1 Part 1: Decision Tree

1.1 Information Gain

Tree with information gain without any pre or post prune gave 0.9467592592592593 accuracy on test set.
The plotted tree can be seen from the file named *tree_inf_gain_nopruning.png*

1.2 Gain Ratio

With gain ratio criterion, I got the acc on test set as 0.9490740740740741
The plotted tree can be seen from the file named *tree_gain_ratio_nopruning.png*

1.3 Average Gini Index

With average gini index, the accuracy is 0.9467592592592593 on the test data.
The plotted tree can be seen from the file named *tree_average_gini_nopruning.png*

1.4 Gain Ratio with Chi-squared Pre-pruning

Using gain ratio as selection criterion and applying chi-squared pre-pruning, the accuracy on the dest dataset is acc 0.9479166666666666.
The plotted tree can be seen from the file named *tree_gain_ratio_prepruning.png*

1.5 Gain Ratio with Reduced Error Post-pruning

With gain ratio and post-pruning, accuracy on the test data is improved and become 0.9502314814814815.
The plotted tree can be seen from the file named *tree_gain_ratio_postpruning.png*

2 Part 2: Support Vector Machine

2.1 First Part

The C parameter informs the optimization of the SVM how much you want to prevent each training example being misclassified. When C values become larger, SVM will choose small margin, which correctly classifies. On the other hand, as can be seen from the plots, if C values are smaller, optimizer will select larger-margin for hyperplane which separates the classes. In this case, it may still be many misclassified examples. Although data is linearly separable, because of the small C value, the classifier will tend to maximize the margin between most of the points, while misclassifying a few points, since the penalty is so low. Therefore it is very likely to get misclassified examples.

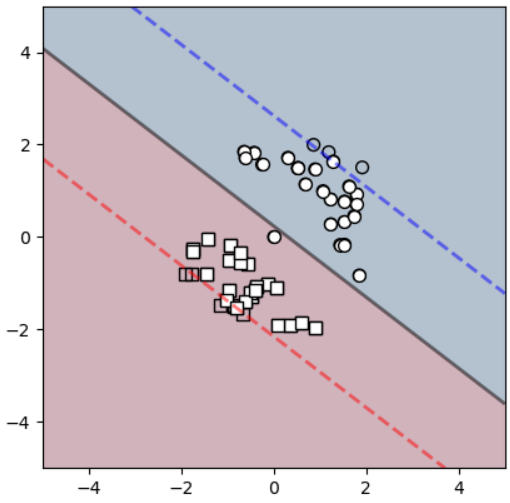

Figure 1: SVM result (linear kernel) and C = 0.01

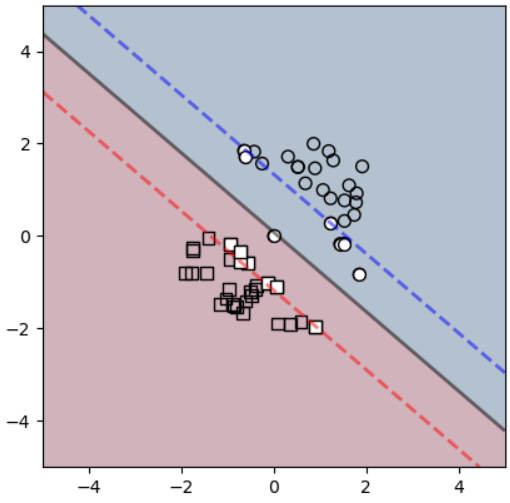

Figure 2: SVM result (linear kernel) and C = 0.1

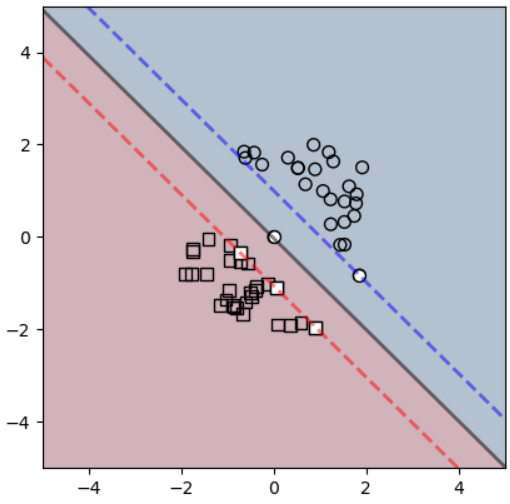

Figure 3: SVM result (linear kernel) and C = 1

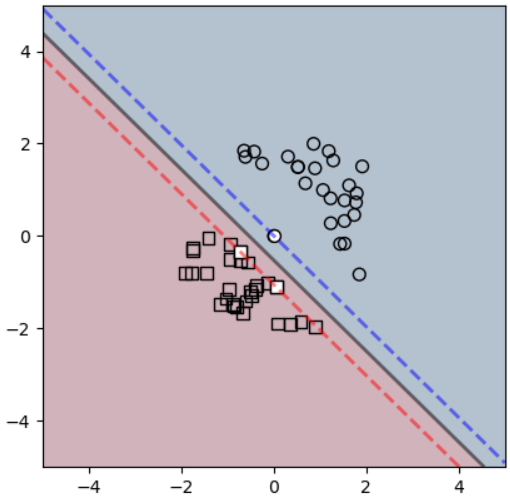

Figure 4: SVM result (linear kernel) and C = 10

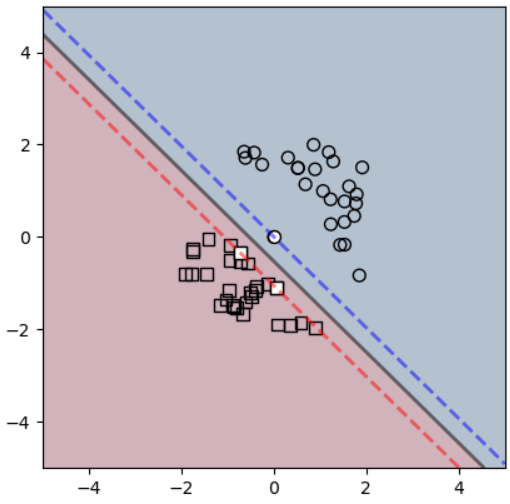

Figure 5: SVM result (linear kernel) and C = 100

2.2 Second Part

Data used in this part is nonlinear. Therefore, I was expecting to get bad results with linear kernel which can be seen from the first plot. Because we use linear kernel when our data is linearly separable, in other words, when it can be separated using a single line.
Note: Explanations are written above each plot.

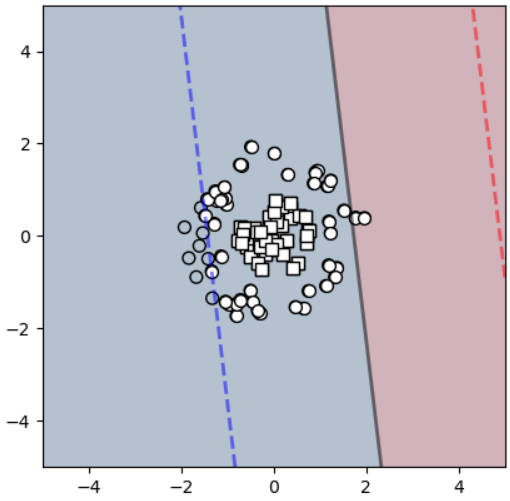

Figure 6: SVM result (linear kernel) and C = 1(default)

When we use polynomial kernel, since the default degree value is 3 for "poly" kernel, instead of linear, because we have additional parameter for hyperplane, rather than straight line, we will get line with curves or some fluctuations. However, this does not perform well, since our data is scattered as two different circles, one is in the inner center, another is in the outer of the centered cluster as a shape of ring.

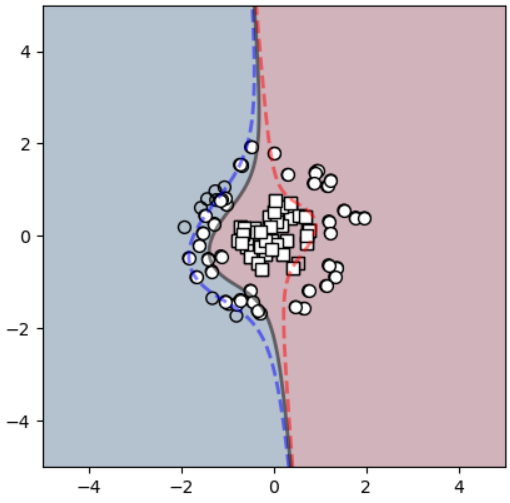

Figure 7: SVM result (poly kernel) and C = 1(default)

Similar to nearest neighbor, RBF kernel also defines similarity as euclidean distance between the two examples. If they are close to each other they get maximum similarity value. Gamma value sets a threshold to decide "when the distance between two example is too far away?". We get good classification boundary because of the higher dimensionality of this trick.

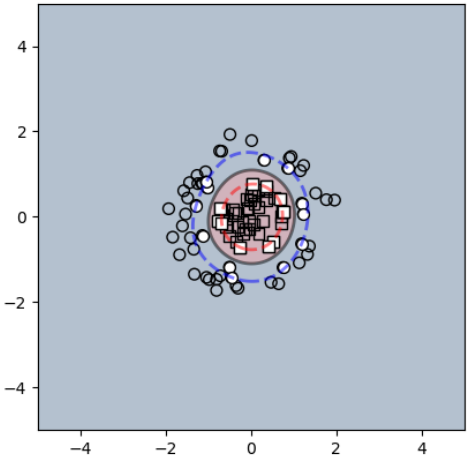

Figure 8: SVM result (rbf kernel) and C = 1(default)

SVM model which uses sigmoid kernel is quite similar to two-layer, perceptron neural network. So, I was expecting to get a better nonlinear separation of the data. However, the reason may be because of the default values of other parameters namely, the intercept constant c, the slope alpha was not good enough and the SVM learned made much complex classification after mapping to high degree space.

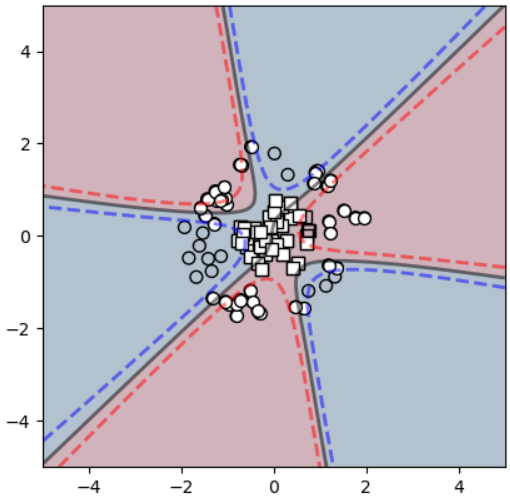

Figure 9: SVM result (sigmoid kernel) and C = 1(default)

2.3 Third Part

The tables below have the example hyperparameters. Feel free the change them. Report the hyperparameters of your best model and its test accuracy.

gamma	C				
	0.01	0.1	1	10	100
-	0.645	0.676	0.708	0.707	0.707

Table 1: Linear kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.538	0.538	0.538	0.607	0.645
0.0001	0.538	0.538	0.610	0.662	0.711
0.001	0.538	0.589	0.678	0.730	0.731
0.01	0.538	0.538	0.712	0.709	0.709
0.1	0.538	0.538	0.710	0.710	0.710
1	0.538	0.538	0.710	0.710	0.710

Table 2: RBF kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.538	0.538	0.538	0.538	0.538
0.0001	0.538	0.538	0.538	0.538	0.542
0.001	0.538	0.542	0.659	0.731	0.725
0.01	0.731	0.725	0.724	0.724	0.724
0.1	0.724	0.724	0.724	0.724	0.724
1	0.724	0.724	0.724	0.724	0.724

Table 3: Polynomial kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.538	0.538	0.538	0.593	0.623
0.0001	0.538	0.538	0.593	0.623	0.640
0.001	0.538	0.591	0.552	0.538	0.523
0.01	0.551	0.526	0.511	0.506	0.505
0.1	0.551	0.520	0.519	0.518	0.519
1	0.547	0.507	0.511	0.513	0.512

Table 4: Sigmoid kernel

Best cross validation result is obtained with parameters 'C': 100, 'gamma': 0.001, 'kernel': 'rbf'
Cross validation accuracy was 0.731. Accuracy on the test set is 0.80.

2.4 Fourth part

2.4.1 Without handling the imbalance problem

Report test accuracy. Can accuracy be a good performance metric? Report confusion matrix and comment on it. Report additional metrics here if you want.

Accuracy is not a good performance metric by itself. Since it may contain misleading information depending on the data whether it is imbalanced or not.

Training with the unchanged train_data, performances measured on the test data according to different metrics:

- accuracy : 0.8333333333333334
- precision_score: 0.8331870061457419

- recall_score: 1.0

As can be seen, accuracy is noticably high for the initial unchanged data. This means that model predicts the value of the majority class for all predictions and achieve a high classification accuracy. I have also calculated other performance metrics for each run to gain meaningful insight information about data. It is important to mention that recall is 1 because recall means "out of all positive examples, how many did I catch?" As can be seen from the confusion matrix, all true data is classified as true which is the ratio 949 / (0 + 949)

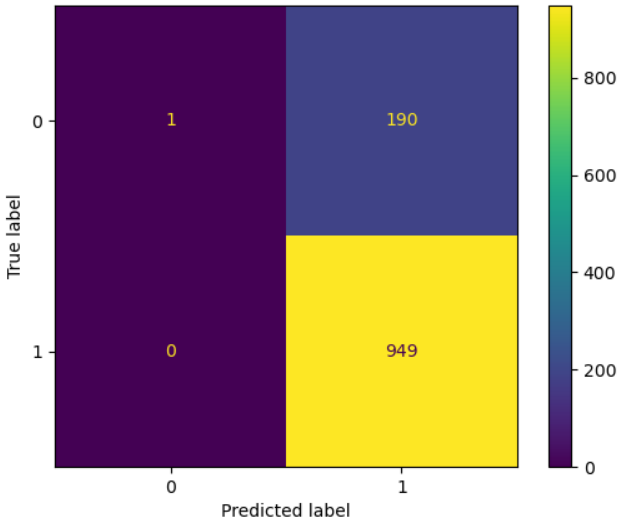

Figure 10: Default Result of catdogimba

2.4.2 Oversampling the minority class

Training with the oversampled train_data, performances measured on the test data according to different metrics:

- accuracy : 0.7956140350877193
- precision_score: 0.8729166666666667
- recall_score: 0.8830347734457323

When oversampled the minority class, we see that accuracy is dropped. This seems like a bad result however, now we have a increased precision which can

be stated as, "our model doesn't say positive to every value, it becomes smarter and ratio of actual positives over all positively labeled examples are increasing.

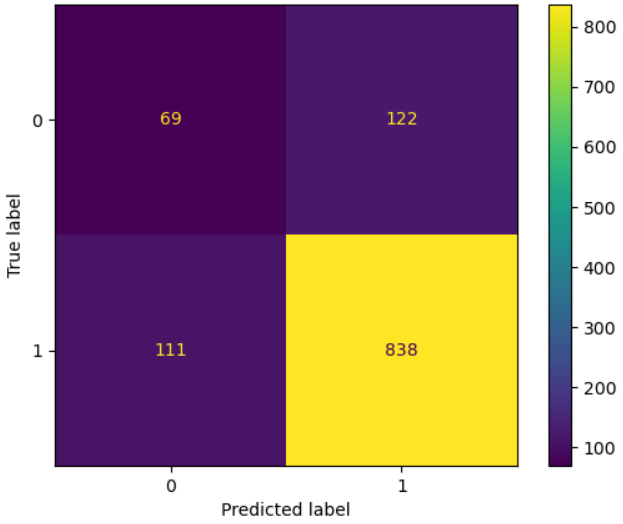

Figure 11: Oversampling Result of catdogimba

2.4.3 Undersampling the majority class

Report your test accuracy, confusion matrix and comment on them.

Training with the undersampled train_data, performances measured on the test data according to different metrics:

- accuracy : 0.5877192982456141
- precision_score: 0.8881685575364667
- recall_score: 0.5774499473129611

When we undersample the data, we see that precision is increased with a significant decrease on recall. This states that model's predicted positive labels are mostly actual positives. Not only the accuracy but also the recall value is dropped, therefore it may not be a good idea to undersample the majority class or maybe it is because of the quality of the remaining or deleted data that made scores low.

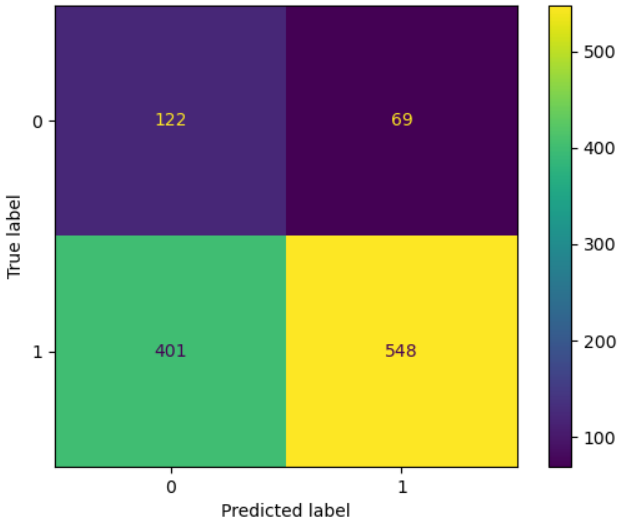

Figure 12: Undersampling Result of catdogimba

2.4.4 Setting the class_weight to balanced

Report your test accuracy, confusion matrix and comment on them.

Training with the balanced train_data, performances measured on the test data according to different metrics:

- accuracy : 0.7578947368421053
- precision_score: 0.8810872027180068
- recall_score: 0.8198103266596417

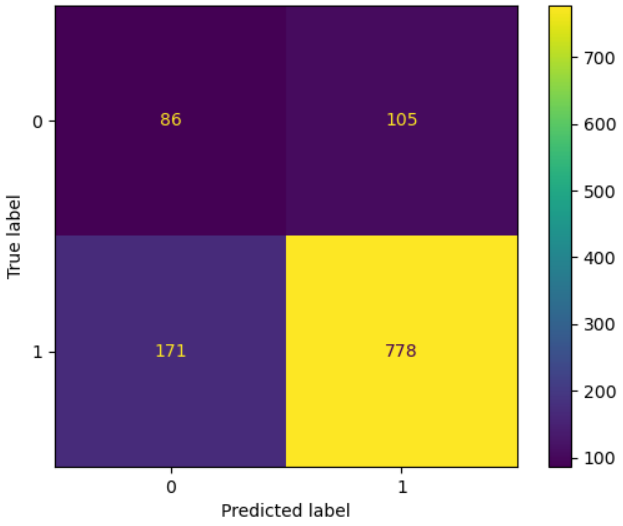

Figure 13: Balanced Result of catdogimba

When we set the class_weight parameter to balanced, we see that precision is increased but the other metrics are decreased. This takes us to the same point as in the undersampling case.