Report

Emil Shikhaliyev

June 13, 2021

1 Part 1: Decision Tree

1.1 Information Gain

Test accuracy for info_gain mode: 0.9333

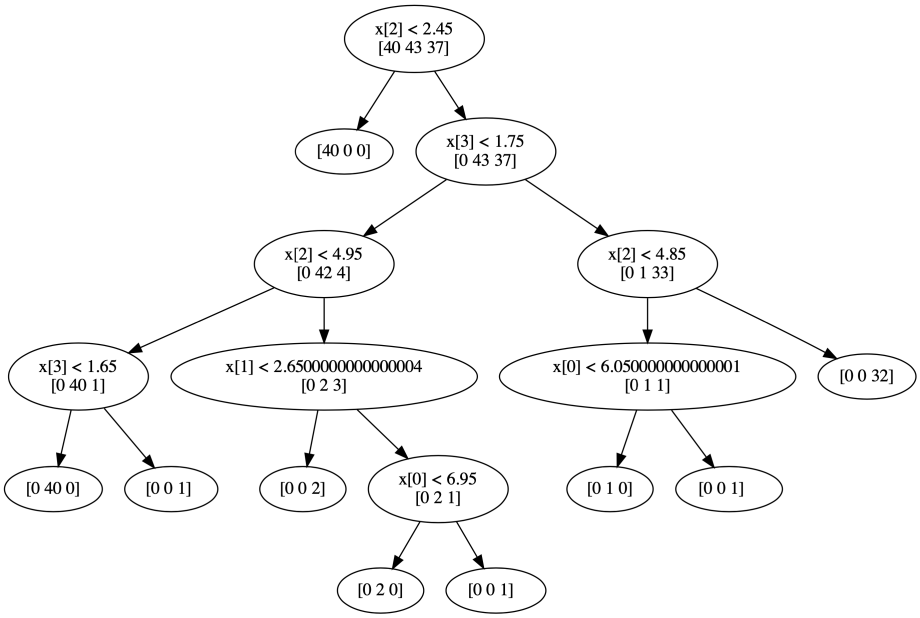


Figure 1: Decision tree for info_gain

1.2 Average Gini Index

Test accuracy for avg_gini_index mode: 0.9

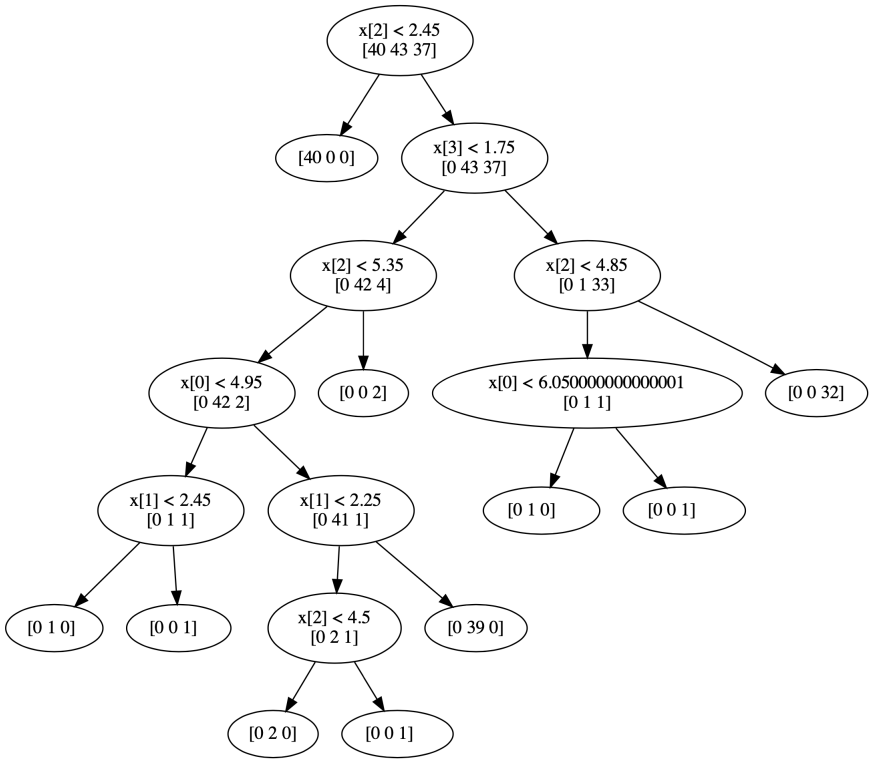


Figure 2: Decision tree for avg_gini_index

1.3 Information Gain with Chi-squared Pre-pruning

Test accuracy for info_gain mode with pre-pruning: 0.96667

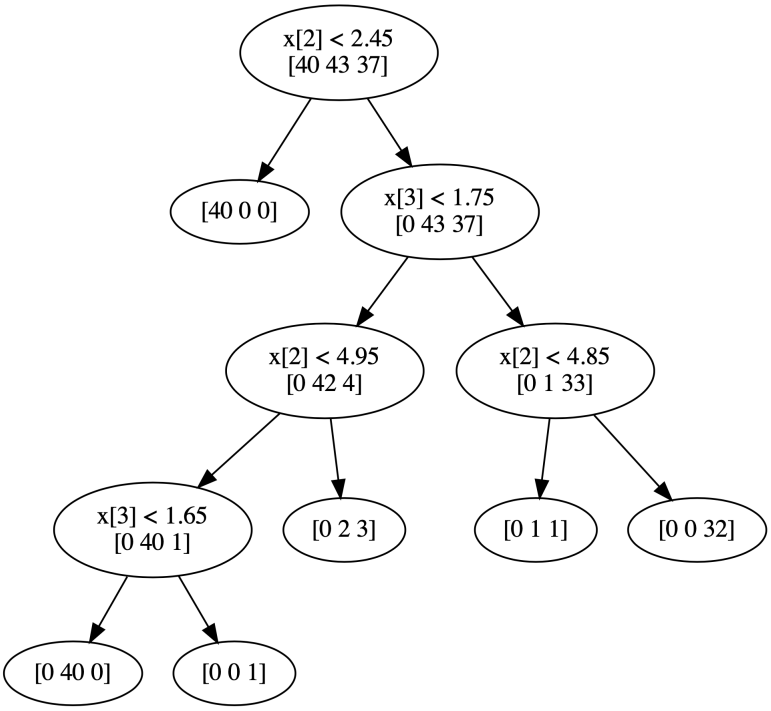


Figure 3: Decision tree for info_gain mode with pre-pruning

1.4 Average Gini Index with Chi-squared Pre-pruning

Test accuracy for avg_gini_index mode with pre-pruning: 0.9

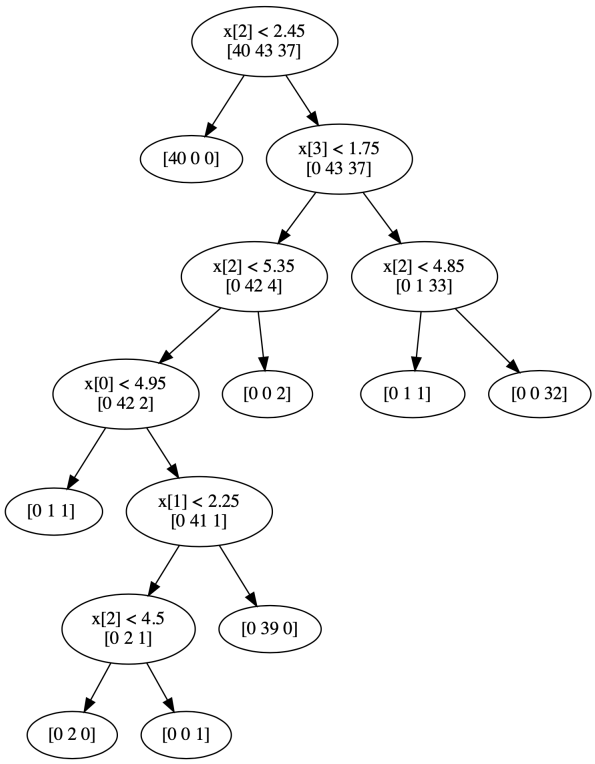
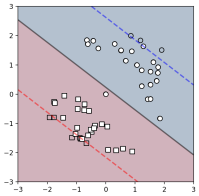


Figure 4: Decision tree for avg_gini_index mode with pre-pruning

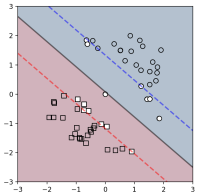
2 Part 2: Support Vector Machine

2.1 First Part

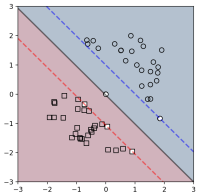
Large C values causes small margin and small C values causes large margin. We can observe that when C value is large enough, most of the instances will be classified correct. But, if we select C values small enough, most of examples will be misclassified.



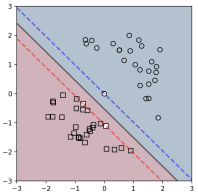
(a) C = 0.01



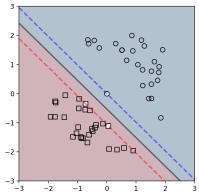
(b) C = 0.1



(c) C = 1



(d) C = 10

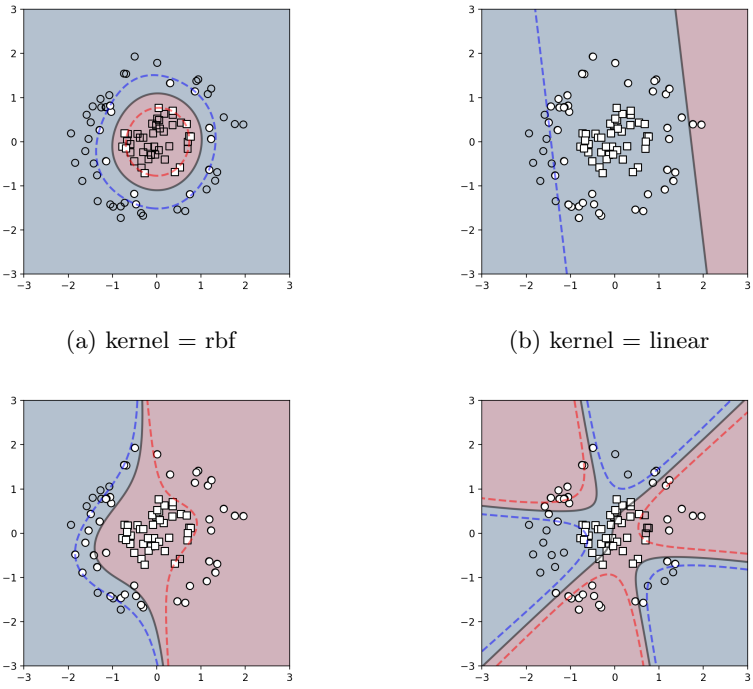


(e) C = 100

Figure 5: Plots of SVMs for each C value

2.2 Second Part

We can get best results with 'rbf' kernel, since the data is not linearly separable.



(a) kernel = rbf

(b) kernel = linear

(c) kernel = polynomial

(d) kernel = sigmoid

Figure 6: Plots of SVMs for each kernel

2.3 Third Part

Best parameters:

$$C = 100, \text{kernel} = \text{rbf}, \text{gamma} = 0.01$$

Test accuracy with these parameters is: **0.88**

gamma	C				
	0.01	0.1	1	10	100
-	0.774	**0.811**	0.779	0.738	0.730

Table 1: Linear kernel

gamma	C				
	0.01.	0.1	1	10	100
0.00001	0.51	0.51	0.51	0.511	0.747
0.0001	0.51	0.51	0.511	0.749	0.796
0.001	0.51	0.511	0.751	0.828	0.857
0.01	0.51	0.761	0.869	0.894	**0.896**
0.1	0.51	0.51	0.883	0.885	0.885
1	0.51	0.51	0.51	0.51	0.51

Table 2: RBF kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.51	0.51	0.51	0.51	0.51
0.0001	0.51	0.51	0.51	0.51	0.51
0.001	0.51	0.51	0.551	0.737	0.81
0.01	0.737	0.81	**0.875**	0.867	0.867
0.1	0.867	0.867	0.867	0.867	0.867
1	0.867	0.867	0.867	0.867	0.867

Table 3: Polynomial kernel

gamma	C				
	0.01	0.1	1	10	100
0.00001	0.51	0.51	0.51	0.51	0.737
0.0001	0.51	0.51	0.51	0.737	0.774
0.001	0.51	0.51	0.736	0.768	**0.787**
0.01	0.51	0.511	0.337	0.687	0.681
0.1	0.51	0.51	0.51	0.51	0.51
1	0.51	0.51	0.51	0.51	0.51

Table 4: Sigmoid kernel

2.4 Fourth part

2.4.1 Without handling the imbalance problem

This model can not understand negative examples accurately. Accuracy is high enough but it does not mean this model works well.

Test accuracy: 0.94933
Recall: 1.00000
Precision: 0.94923

Predicted	True Condition	
	+	-
+	1421	76
-	0	3

Table 5: Confusion Matrix

2.4.2 Oversampling the minority class

Since I increased the count of data points, which are belonging to minority class. Now our new data list have more negative examples than train_data. Count of negative and positive examples are same. So, we can observe that model can understand more negative examples than model in previous part.

Test accuracy: 0.958
Recall: 0.98944
Precision 0.96699

Predicted	True Condition	
	+	-
+	1406	48
-	15	31

Table 6: Confusion Matrix

2.4.3 Undersampling the majority class

Since I decreased the count of data points, which are belonging to majority class. Now our new data list have less positive examples than train_data. Count of negative and positive examples are same. So, we can observe that model can understand more negative examples than model in previous part. But, test accuracy is less than previous parts.

Test accuracy: 0.81667
Recall: 0.82688
Precision: 0.97591

Predicted	True Condition	
	+	-
+	1175	29
-	246	50

Table 7: Confusion Matrix

2.4.4 Setting the class_weight to balanced

In this part I used SVC(class_weight='balanced'). So this command arranges the weight of classes according their frequencies. So, this model can understand negative examples more clearly.

Test accuracy: 0.95867
Recall: 0.98522
Precision: 0.97155

Predicted	True Condition	
	+	-
+	1400	41
-	21	38

Table 8: Confusion Matrix