CEN G499

CENG499
INTRODUCTION TO MACHINE LEARNING
THE3
REPORT

Erdem
Şamlıoğlu
2448843

1 Part1

Gain Ratio Criterion

Selected features in order and with their Gain Ratio values.

- **milk** (Gain: 1.0)
- **fins** (Gain: 1.0000000000000002)
- **feathers** (Gain: 0.9999999999999999)
- **backbone** (Gain: 0.9999999999999999)
- **airborne** (Gain: 0.6073508754546144)
- **predator** (Gain: 0.3448660840774643)
- **legs** (Gain: 1.0)
- **tail** (Gain: 0.6073508754546144)
- **aquatic** (Gain: 0.3448660840774643)
- **eggs** (Gain: 1.0)

Information Gain Criterion

Selected features in order and with their Information Gain values.

- **legs** (Gain: 1.3630469031539394)
- **fins** (Gain: 0.8865408928220899)

- **toothed** (Gain: 0.9852281360342515)
- **eggs** (Gain: 0.6962122601251458)
- **hair** (Gain: 0.8256265261578954)
- **hair** (Gain: 0.6892019851173654)
- **aquatic** (Gain: 0.8631205685666308)
- **toothed** (Gain: 0.7219280948873623)
- **aquatic** (Gain: 0.7219280948873623)

2 Part2

2.1 Dataset 1

For Dataset 1, I have used 4 different configurations(C: 1 and 10, kernel:linear and rbf)

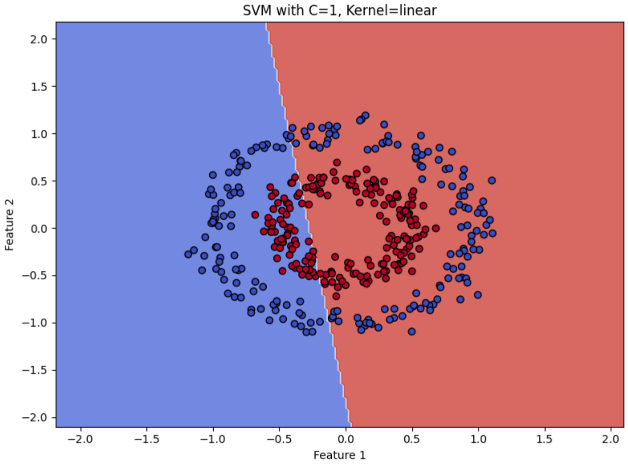

Figure 1: SVM with $C = 1$, Kernel=Linear

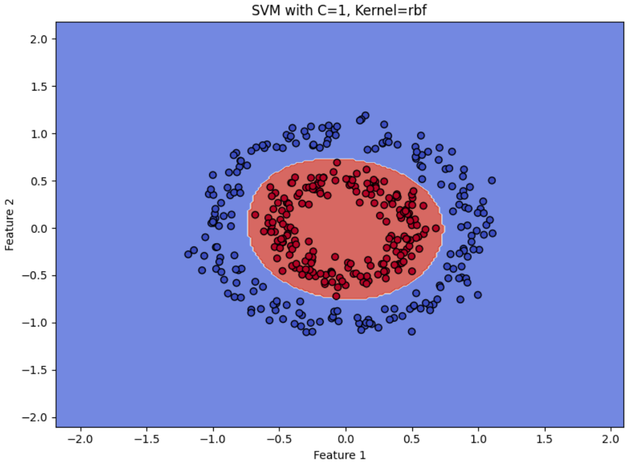

Figure 2: SVM with $C = 1$, Kernel=RBF

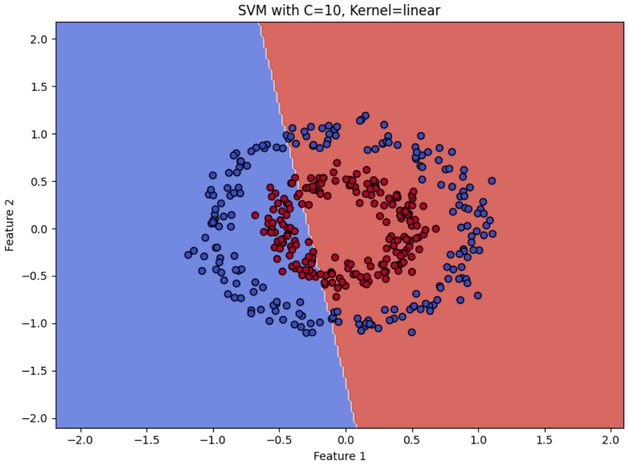

Figure 3: SVM with $C = 10$, Kernel=Linear

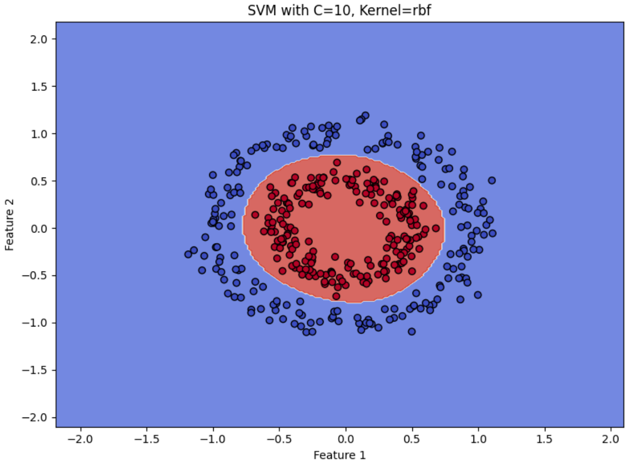

Figure 4: SVM with $C = 10$, Kernel=RBF

2.2 Dataset 2

Mean Accuracy results of the 5 repetitions of 10 fold cross validation with 4 different configurations:

Configuration	C	Kernel	Mean Accuracy	95% Confidence Interval
1	1	Linear	0.748	[0.737, 0.759]
2	1	RBF	0.763	[0.754, 0.772]
3	10	Linear	0.748	[0.737, 0.759]
4	10	RBF	0.753	[0.743, 0.764]

Table 1: Configurations' Mean Accuracy Results

The best configuration with the highest mean accuracy is Kernel:RBF and C:1. Considering the confidence intervals are overlapping and mean accuracies are really close, all these four configurations are similar in performance but as it is stated RBF kernel and C:1 is slightly beter in mean accuracy metric.

3 Part3

Algorithm	**Most-Chosen Hyperparameters**	**Times Chosen**	**Mean F1**	**95% CI**
KNN	{n_neighbors=3, metric=manhattan}	15/15	0.8552	[0.8510, 0.8594]
SVM	{C=1.0, kernel=rbf}	15/15	0.8279	[0.8223, 0.8335]
Decision Tree	{max_depth=10, criterion=entropy}	8/15	0.7982	[0.7894, 0.8070]
Decision Tree	{max_depth=10, criterion=gini}	7/15		
Random Forest	{max_depth=10, n_estimators=100}	15/15	0.8633	[0.8578, 0.8688]
MLP	{hidden_layer_sizes=(100,), alpha=0.001}	6/15	0.8518	[0.8480, 0.8556]
MLP	{hidden_layer_sizes=(100,), alpha=0.0001}	9/15	0.8518	
Gradient Boosting	{n_estimators=100, learning_rate=0.1}	15/15	0.8482	[0.8422, 0.8543]

Table 2: Nested Cross Validation F1 results and which hyperparameters are selected.

Hyperparameter configurations used:

- **Hyperparameter Configurations:**
  - KNN: {n_neighbors=[3,5], metric=[*euclidean*, *manhattan*]}
  - SVM: {C=[0.1,1.0], kernel=[*linear*, *rbf*]}
  - Decision Tree: {max_depth=[5,10], criterion=[*gini*, *entropy*]}
  - Random Forest: {n_estimators=[50,100], max_depth=[5,10]}
  - MLP: {hidden_layer_sizes=[(50,),(100,)], alpha=[1e-3,1e-4]}
  - Gradient Boosting: {n_estimators=[50,100], learning_rate=[0.01,0.1]}

I used random_state to eliminate randomization.

As shown, Random Forest obtains the highest mean F1 with 0.8633 with 95% CI [0.8578, 0.8688], also KNN with 0.8552 and MLP with 0.8518 are close to that. For certain algorithms, SVM, KNN, Random Forest, chose the same parameter set for all folds, showing a strong hyperparameter combination(among the ones I have tested) among others.
Decision Tree chose different criteria, gini and entropy, time to time and ended up with a lower overall F1.
Gradient Boosting showed good performance but it took much longer in runtime compared to other methods.(In my PC it took 12-13 hours to run whole methods, Gradient Boosting took 9-10 hours itself. Later I tried it in Google Colab, It took 2-2.5 hours and Gradient Boosting took around 1.5 hours.)

In conclusion, for this ECG dataset, Random Forest yields the best average F1 among the tested methods and hyperparameter grids.