SCALABLE MACHINE LEARNING

Assignment 2

RAMEEZ HUSSAIN
180128228

Introduction

For this assignment I have attached the following files alongside this report:

- Q1_1.py, Q1_1_Script.sh and Q1_1_Output.output, this is the code, bash and output file for question 1 part 1
- Q1_2.py, this is the code file for question 1 part 2 and 3
- Q1_2_10_Cores_Script.sh and Q1_2_10_Cores_Output.output, this is bash and output file of Q1_2.py ran with 10 cores
- Q1_2_20_Cores_Script.sh and Q1_2_20_Cores_Output.output, this is bash and output file of Q1_2.py ran with 20 cores

Question 1

This question was concerning the search for high-energy physics using certain classic supervised learning algorithms. I had to train classification algorithms over the data to identify Higgs bosons from particle collisions. A Random Forest and Gradient Boosting models were trained over the dataset to identify the Higgs bosons.

1.1 Choosing the best configurations of parameters

The different algorithms I have implemented have various options for parameters and finding the best parameters can help in increasing the performance of the model. To find out the best parameters for each algorithm I used CrossValidator which enables the coder to set various and different options for the model, using a ParamGridBuilder to set different options for each model and Pipeline to combine multiple algorithms. I have used the CrossValidator for a small part of the data 5%. The configurations that I choose and tested are as follows:

- **Random Forest Classifier:** maxDept: 5,10 and 15 and numTrees: 3,10 and 20
- **Gradient Boosting Classifier:** maxDept: 5,10 and 14 and maxIter: 5,10 and 15

The results are displayed in Table 1 below:

Model	Area Under the ROC curve	Accuracy
Random Forest Classification	0.7179459196783484	0.7197609145420932
Gradient Boosting Classification	0.7177211405576462	0.7190115648351069

Table 1: Best configurations of parameters and metric scores

After I have tested my code, I achieved the following best parameters:

- **Random Forest Classifier:** maxDept: 15 and numTrees 20
- **Gradient Boosting Classifier:** maxDept: 10 and maxIter: 15

To achieve the results for AreaUnderROC and Accuracy a BinaryClassificationEvaluator and MulticlassClassificationEvaluator was used and a 7:3 ratio for training and testing was used.

1.2 Comparing classification algorithms

After I have received the results from above, I applied the best parameter configurations in each individual algorithm to classify the particles, this was done for the full dataset. To achieve this, I used 10 cores and 20 cores of the system, AreaUnderROC, Accuracy, and timings were noted.

The results are displayed in Table 2 below:

Model	Area Under the ROC curve	Accuracy	Timing for 10 Cores	Timing for 20 Cores
Random Forest Classification	0.7271346824708538	0.7290239065620371	29 mins 31 secs	29 mins 16 secs
Gradient Boosting Classification	0.7244831322616015	0.7259357571573144	11 mins 34 secs	11 mins 10 secs

Table 2: Metric scores for the full dataset

1.3 Most Important features

After I completed the steps above now, I had to find the three most relevant features for each model. Please note that that c23 refers m_jjj, c25 refers to m_jlv, c26 refers to m_bb, c27 refers to m_wbb and c28 refers to m_wwbb.

The results are displayed in Table 3 below:

Model	Relevant feature 1	Relevant feature 2	Relevant feature 2
Random Forest Classification	_c26	_c28	_c27
Gradient Boosting Classification	_c26	_c23	_c25

Table 3: Most relevant features

1.4 Any interesting observations

An interesting observation regarding the top features question is that both models produce very different results. Only m_bb appears in both models and the others (m_jjj, m_jlv, m_wbb, and m_wwbb) appear only in one of the respected models.

We can notice that from the question comparing classification algorithms for the full dataset that 20 Cores run faster compared against 10 cores. The results for both models seem to be quite similar and the Gradient Boosting algorithm runs significantly faster compared against Random Forest which takes around 11 minutes.

It is very important to note that for the small dataset (5%) that both models produce very similar results. However, if you pay close attention it can be seen that the Gradient Boosting model has slightly lower results so it can be assumed that the Random Forest model performance and results are slightly better.