Report: HomeWork_01

Federico Kieffer
(n.m.1722271)

The present work focuses on the solution of a Machine Learning classification problem. In particular, the problem of compiler provenance has been tackled in a double fashion: a binary classifier that can predict if a function has been compiled with optimization 'High' or 'Low' has been implemented, as well as a multiclass classifier that can predict the compiler used to compile a specific function.
To avoid computational inefficiency of the implemented algorithms, just 3000 samples of the whole given dataset have been taken into account.

PROBLEM 1:

At the feature extraction stage, some different variants have been explored. In particular, since the original dataset is unbalanced with respect to the optimization label, both a balanced and an unbalanced subsets of 3000 samples have been considered (ref. Code "Problem1", Sections "Generate lists of samples and labels" and "Balance the dataset"). Moreover, in both these two cases, three feature extraction variants have been investigated. Firstly, the "importance" of the words of the full instructions has been considered. Secondly, the "importance" of only the single mnemonics of each instruction have been taken into account (ref. Code "Problem1", Section "Generate list of Mnemonics") and finally the "importance" of only a certain number (20) of initial and final mnemonics have been used as input for the classification problem (ref. Code "Problem1", Section "Generate cut list of mnemonics").

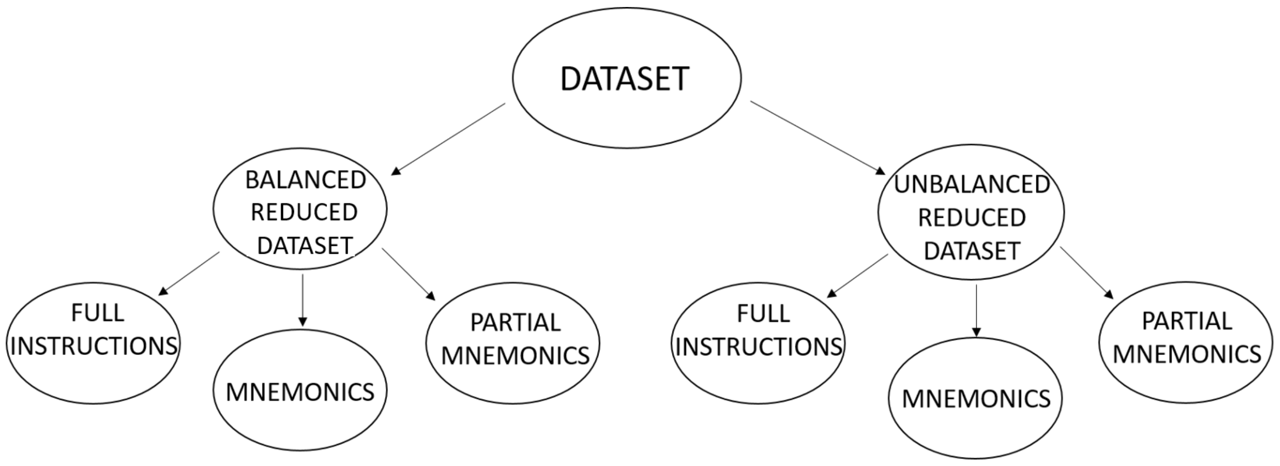
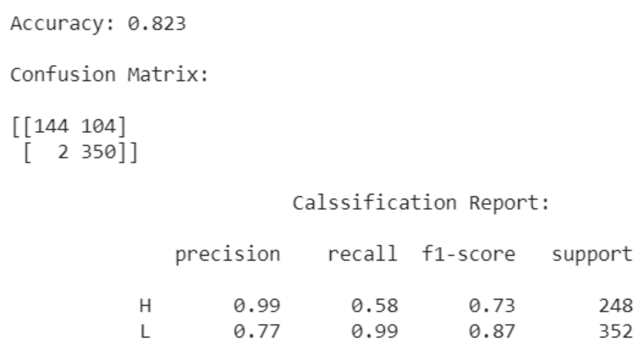


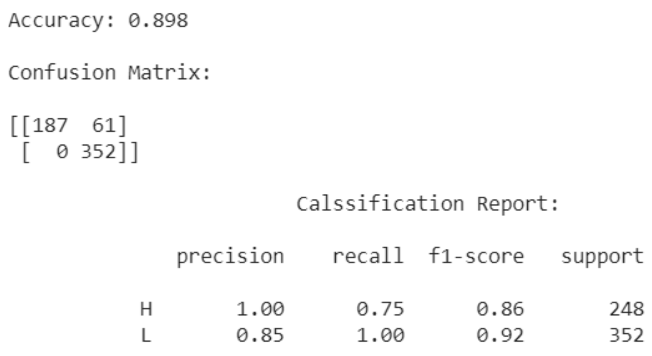
Figure 1: Feature Extraction Variants for the binary classification problem

In particular, the distinction in balanced and unbalanced datasets has been performed in view of the application of a Multinomial Naive Bayes (MNB) model, whose classification performances are expected to worsen as the samples used in the training phase gets unbalanced. On the other hand, the remaining steps of decision flow adopted to solve the problem have been kept always the same. In particular, since the samples in the dataset are made of lists of instructions, it was needed first of all to generate a rapresentation of the dataset that can be provided as input for the machine learning algorithm. This operation has been carried out via a vectorization of the instructions. More precisely, each word of each sample has been associated to a number that quantifies the "importance" of that word. This has been done using the tfidfVectorizer. Moreover, in order to enhance the performances of the resulting classifier, the vectorization has been performed using N-grams. More precisely, in addition to using each individual word found in the instructions, also 2-grams, 3-grams and up to 7-grams have been used as features. Finally, a binarization of the tf term has been also performed. (ref. Code "Problem1", Section ''Feature Extraction''). After the feature extraction phase, the dataset has been randomly partitioned in training and test set. In particular, the dimention of the test set has been chosen as the 20% of all the used samples. The remaining 80% of the available samples have been used to train a MNB model. The choice of splitting the data in training and test sets is crucial for evaluation purposes. More precisely, it is fundamental not to exploit all the available labeled samples in the training phase so as to use some of them in a later evaluation phase. Moreover, the split in training and test sets has been performed randomly so as to mantain on both these two resulting sets the same distribution that characterizes the dataset itself. Here follow the classification performances obtained using the unbalanced dataset for the three feature extraction variants explored:



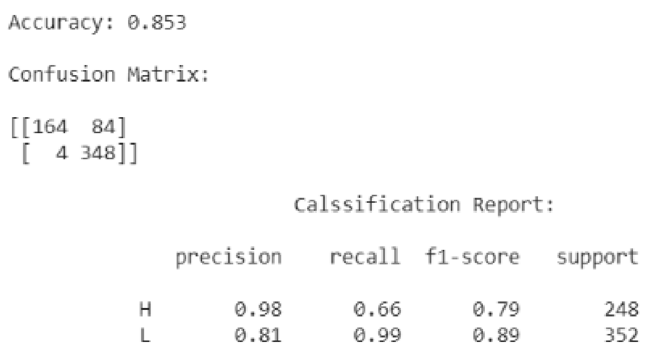
```
Accuracy: 0.823

Confusion Matrix:

[[144 104]
 [  2 350]]

                  Calssification Report:

              precision    recall  f1-score   support

           H       0.99      0.58      0.73       248
           L       0.77      0.99      0.87       352
```

Figure 2: Full Instructions (Unbalanced)



```
Accuracy: 0.898

Confusion Matrix:

[[187  61]
 [  0 352]]

                  Calssification Report:

              precision    recall  f1-score   support

           H       1.00      0.75      0.86       248
           L       0.85      1.00      0.92       352
```

Figure 3: Mnemonics (Unbalanced)

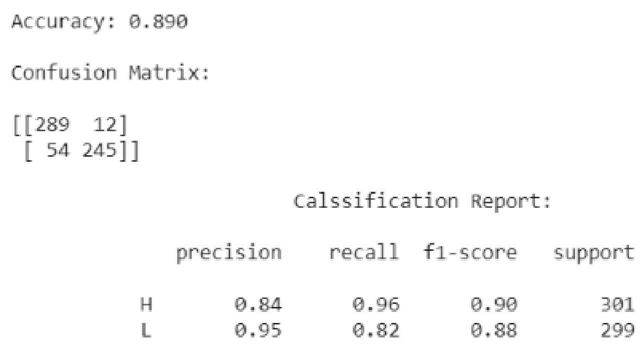


```
Accuracy: 0.853

Confusion Matrix:

[[164  84]
 [  4 348]]

                  Calssification Report:

              precision    recall  f1-score   support

           H       0.98      0.66      0.79       248
           L       0.81      0.99      0.89       352
```

Figure 4: Partial Mnemonics (Unbalanced)

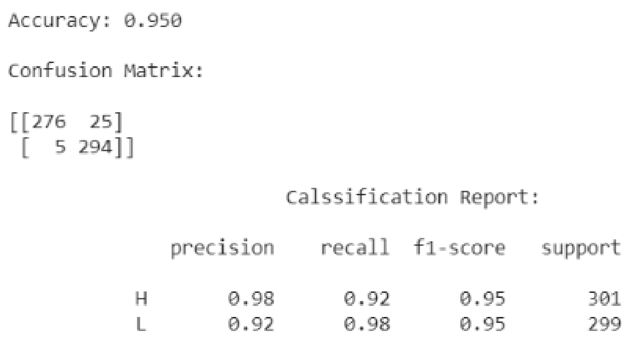
The best classifier among the three is the one that has been trained using only the mnemonics of each instruction as features. In fact, it provides the highest accuracy and, more precisely, it is characterized by the highest capability of avoiding false positives and false negatives, which is highlighted by both a more elevated precision and recall. A remarkable defect of this solution is the relatively low recall associated to the class "H", which is indeed reflected into a quite high number of false negatives (61), as reported into the associated confusion matrix. On the other hand, the performances associated to the classifier resulting from the use of only a part of the total number of mnemonics are worse both in terms of accuracy and f1-score (i.e. the harmonic mean of precision and recall). However, this machine learning (ML) solution still provides more satisfactory results than the ones generated by the classifier trained on the full instructions, whose perfromances are the poorest in terms of all the performance metrics above considered.

As reported in the following, the use of a balanced dataset in the training phase yields machine learning solutions with more satisfactory performances for each of the three feature extraction cases previously mentioned:



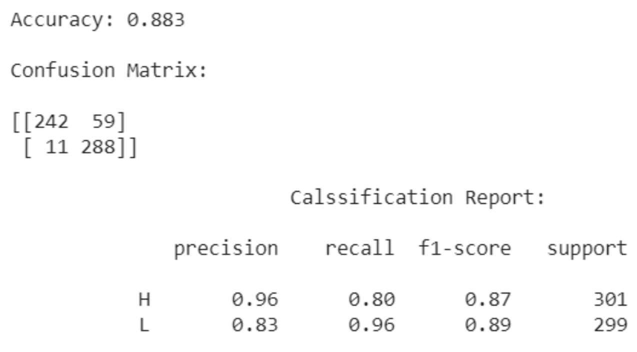
```
Accuracy: 0.890

Confusion Matrix:

[[289  12]
 [ 54 245]]

                     Calssification Report:

           precision    recall  f1-score   support

H               0.84      0.96      0.90       301
L               0.95      0.82      0.88       299
```

Figure 5: Full Instructions (Balanced)



```
Accuracy: 0.950

Confusion Matrix:

[[276  25]
 [  5 294]]

                     Calssification Report:

           precision    recall  f1-score   support

H               0.98      0.92      0.95       301
L               0.92      0.98      0.95       299
```

Figure 6: Mnemonics (Balanced)



```
Accuracy: 0.883

Confusion Matrix:

[[242  59]
 [ 11 288]]

                     Calssification Report:

           precision    recall  f1-score   support

H               0.96      0.80      0.87       301
L               0.83      0.96      0.89       299
```

Figure 7: Partial Mnemonics (Balanced)

Once again, the best-performing classifier is the one trained on mnemonics only. In particular, with respect to the unbalanced case, this solution is characterized by a significant rise of accuracy and by a much higher capacity of avoiding false negatives. From the analisis of the performance metrics associated to the other two ML solutions, a remarkable occurrence can be pointed out. Although the two

classifiers are characterized by a very similar accuracy, they behave in a quite different way. In fact, by direct inspection of the respective confusion matricies, it can be observed that the solution associated to the full instructions is abler to avoid false negatives, whereas the one obtained using only a part of the mnemonics has a much higher capability of avoiding false positives. This result highlights the fact that often the accuracy may not be enough to fully assess the performances of a classifier.
In conclusion, from the comparison of the three feature extraction alternatives illustrated, it turns out that the most significant feature for the classification of the optimization level is rapresented by the mnemonics. Moreover, as the dataset becomes unbalanced, a general trend can be also worked out since the performances of the resulting classifiers tend to worsen in all the different cases explored. As above mentioned, this result was quite expected once a MNB model had been chosen.

PROBLEM 2:

Unlike in the binary classification problem, the multiclass problem is provided with an already balanced dataset and therefore no distinction has been performed from this particular point of view. However, since only a reduced part of the dataset has been taken into account, a particular attention has been paid in making sure that this extracted sub-dataset was balanced too. More precisely, it has been made sure that the reduced dataset had an equal number of samples for all the three classes (ref. Code "Problem2", Section "Consider limited part of the data"). The same three different feature extraction variants of the previous problem have been here investigated too.

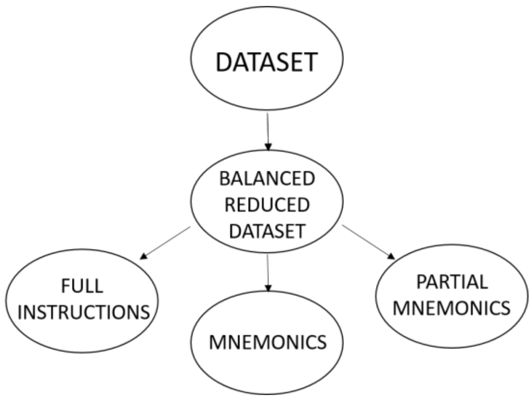
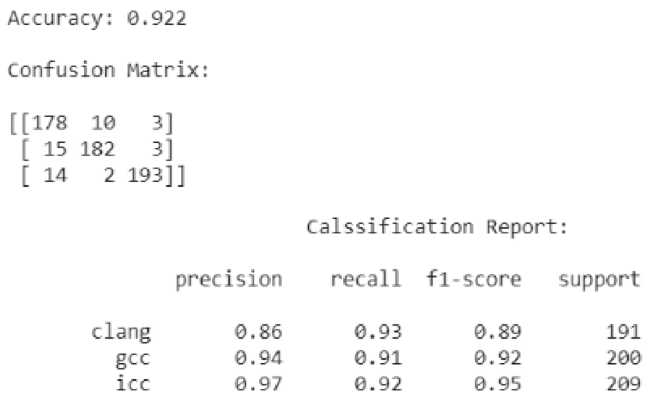


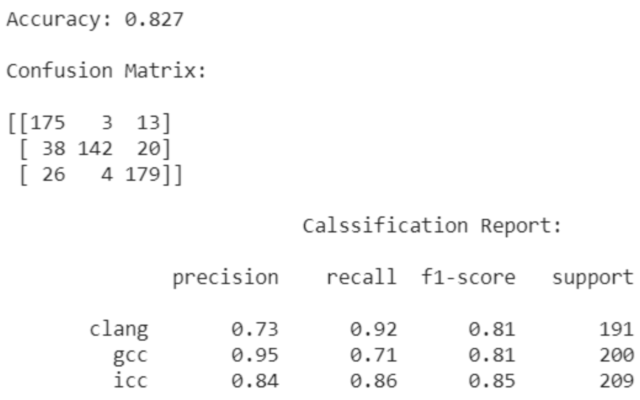
Figure 8: Feature Extraction variants for the multiclass problem

Also the remaining steps of the decision flow adopted in the first problem have been kept the same: the information contained in the dataset has been encoded using the tfidf vectorizer, the training of a MNB model has been carried out using 80% of the available samples and the prediction performances of the resulting classifier have been assessed thanks to the remaining 20% of samples. Here follow the performances obtained in the three different feature extraction cases:



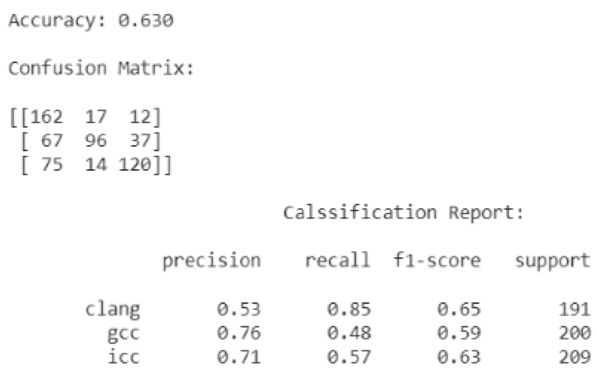
```
Accuracy: 0.922

Confusion Matrix:

[[178  10   3]
 [ 15 182   3]
 [ 14   2 193]]

                    Calssification Report:

           precision    recall  f1-score   support

   clang       0.86      0.93      0.89       191
     gcc       0.94      0.91      0.92       200
     icc       0.97      0.92      0.95       209
```

Figure 9: Full Instructions



```
Accuracy: 0.827

Confusion Matrix:

[[175   3  13]
 [ 38 142  20]
 [ 26   4 179]]

                    Calssification Report:

           precision    recall  f1-score   support

   clang       0.73      0.92      0.81       191
     gcc       0.95      0.71      0.81       200
     icc       0.84      0.86      0.85       209
```

Figure 10: Mnemonics



```
Accuracy: 0.630

Confusion Matrix:

[[162  17  12]
 [ 67  96  37]
 [ 75  14 120]]

                    Calssification Report:

           precision    recall  f1-score   support

   clang       0.53      0.85      0.65       191
     gcc       0.76      0.48      0.59       200
     icc       0.71      0.57      0.63       209
```

Figure 11: Partial Mnemonics

It is evident that the best classifier is in this case the one trained using the full instructions list. In particular, not only it exhibits a much higher accuracy, but it is also characterized by a greater precision and recall with respect to all the three classes involved. This last result is confirmed by the much lower number of off-diagonal terms in the corresponding confusion matrix. However, this best-performing classifier has still as most remarkable defect the tendecy of generating false positives with respect to the "clang" class. This phenomena may be caused by a non-perfectly balanced training set, which is indeed characterized by a slightly higher number of samples labeled as "clang" (as highlighted by the fact that the corresponding number of samples of the test set labeled as "clang" is lower). As support of what just mentioned, also the remaining two classifier are affected by the same kind of pathology. On the other hand, using only the 20 inital and final mnemonics as features yields in this case the worst classifier in terms of all the performance metrics considered.
To conclude, the most significant feature for the classification of the compiler provenance seems to be the whole list of instructions.

CONCLUSIONS:

Two different compiler provenance problems have been tackled. In both cases, different feature extraction variants have been explored and the performances of the corresponding MNB models have been assessed. In particular, accuracy, precision, recall and f1-score have been considered as most significative performance metrics. The associated confusion matricies have been also derived. In the first case, the best-performing classifier has been obtained using only the "importance" of each mnemonics as features, while in the second problem the most satisfacory results have been achieved by feeding the vectorizer with the entire instructions.
This case study highlights the fact that, depending on the specific function that the problem requires to learn, an appropriate choice of the features can lead to much better performances of the resulting classifier.