DEPARTMENT OF COMPUTER, CONTROL, AND MANAGEMENT ENGINEERING ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Machine learning

Report Homework 1

Filippo Betello

1835108

SUMMARY

CHAPTER 1: PROJECT OVERVIEW

In this homework the goal is to provide a solution for a binary classification problem described in the seminar *"Code model for secure and debuggable systems"*.

1.1 Dataset

It is given one type of a pandas data frame separated by *tab* containing 100K samples: each sample contains three type of information:

1. List of assembly instructions
2. Source Line
3. A label that indicates whether the mapping is correct or not

The dataset is balanced and there are no duplicates.

The first operation is to split each line in *instructions, source_line, bug*. The next step is to merge the variable *data* with the strings of *instructions and source_line*. By doing this, the machine learning algorithm has much more relevant information.

CHAPTER 2: PRE-PROCESSING

Data pre-processing is a fundamental step in every machine learning problems and more because it affects the final output. It is necessary to transform the data into vector representation, doing this is possible to apply machine learning algorithm. There are different types of data-processing: in this homework two of them are presented. To use them it is necessary to write this line of code

from sklearn.feature_extraction.text import *

2.2 Method 1: Hashing Vectorizer

The first method presented is the Hashing Vectorizer, that correspond to a multivariate distribution. The main advantage to use this vectorization is the memory scalability: there is no need to store a vocabulary dictionary in memory. In addition, it is possible to set the *stop_words* parameter, but in this exercise it is not relevant because it is impossible to know in advance what is the cause of the bug, so everything could be important. On the other hand, once vectorized the features' names can no longer be retrieved.

2.3 Method 2: Tfidf Vectorizer

The second method presented is the Tfidf vectorizer. It is almost equivalent to the Count Vectorization (multinomial distribution), but it is followed by a transformation. The count vectorization is used to transform the input into a vector based on the frequency (count) of each word that occurs in the document. Then transform the count matrix into a normalized tf (term frequency) or td-idf(term frequency times inverse document frequency) representation.

CHAPTER 3: SPLIT DATA

For both method 1 and method 2, it is mandatory to use the function *train_test_split* to split the dataset. The parameters of this function are:

1. X_all that represents the result of the corresponding vectorization with the function *fit_transform* passing *data* (previously defined) as variable;
2. y_all that takes the *bug* label from the dataset;
3. *test_size* allows me to decide the size of data that will be used as the test set. In this problem, the data was split as follows: 20% test and 80% training, which is the general rule;
4. *random_state* if set as an integer is important to keep the same split during time.

CHAPTER 4: EVALUATION METRICS

To compare the two methods, the results obtained are presented with Classification report and Confusion matrix.

4.1 Classification Report

4.1.1 ACCURACY

Accuracy can show immediately if a model is being trained good or not, and how it could perform generally. However, sometimes, it does not give all the information about a problem. The accuracy is given by:

$$Accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions}$$

For a binary classification problem, as in this case, the accuracy can also be calculated as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Where:

TP = True positive;

TN = True negative;

FP = False positive;

FN = False negative.

4.1.2 PRECISION

The precision (also known as positive predicted value) helps when the count of false positive is high:

$$Precision = \frac{True\ positive}{Predicted\ positive} = \frac{TP}{TP+FP}$$

4.1.3 RECALL

The recall (also known as sensitivity) is helpful when the number of false negative is high:

$$Recall = \frac{True\ positive}{Real\ positive} = \frac{TP}{TP+FN}$$

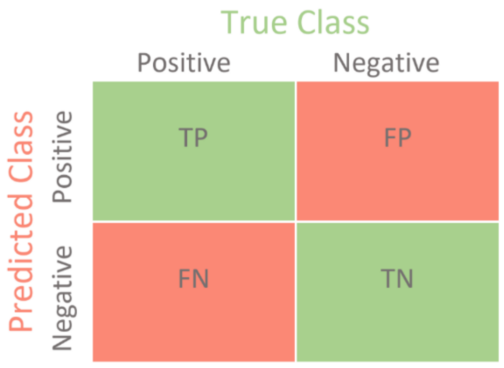
4.1.4 F1-SCORE

F1 score is the harmonic mean of precision and recall and is a measure of test's accuracy:

$$F1\ score = 2 * \frac{precision * recall}{precision+recall}$$

4.2 Confusion matrix

Confusion matrix is a specific table that helps us to visualize the performance of an algorithm. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class and vice-versa. The main diagonal represent the accuracy for each class. In a binary classification problem, the matrix will be *2x2*: it will report the number of *true positive, true negative, false positive, false negative* as in the figure:



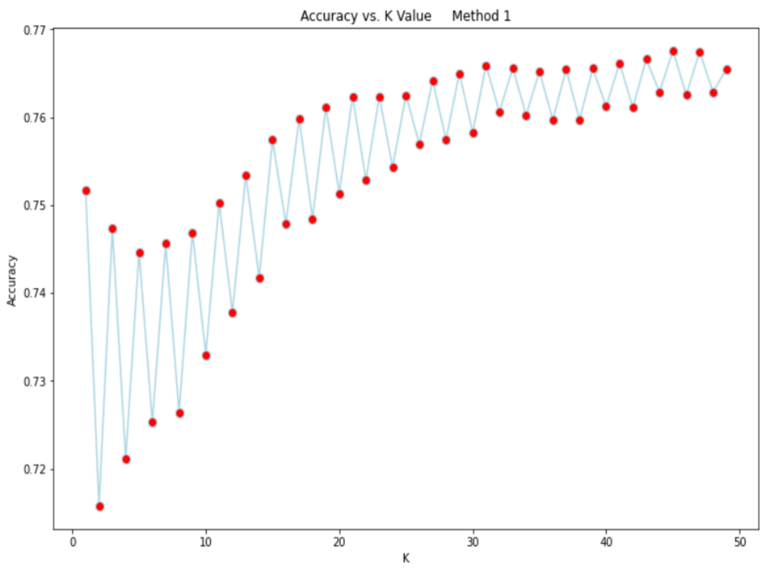
CHAPTER 5: MODELS

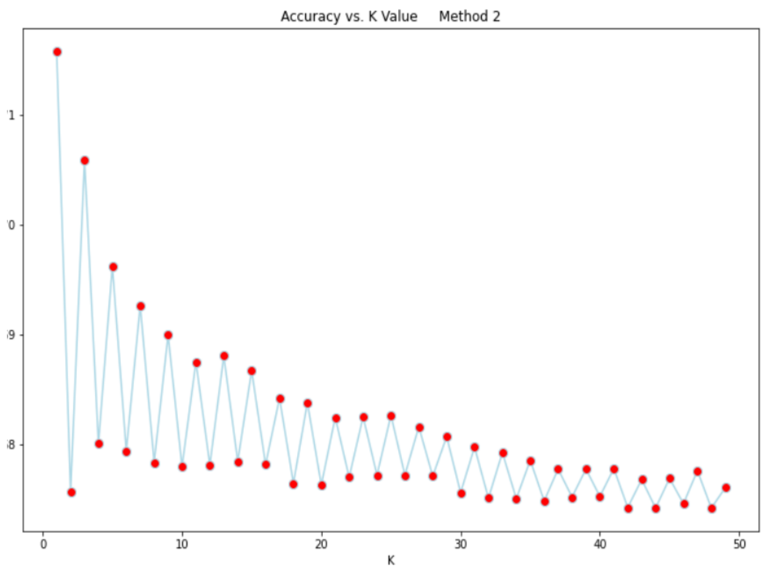
Four different models are used to test the performances:

1. **Bernoulli** is a naïve bayes classifier for multi-variate Bernoulli models. It is designed for binary/boolean feature. To use it, it is necessary to import *sklearn.naive_bayes*.

2. **Logistic regression** fits a linear model to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation. It is mostly used when the data has binary output.

3. **K-nearest neighbors** is an algorithm where the classification is computed from a simple majority voting of the nearest neighbors of each point. To compute the best value of K, it is plotted the accuracy and K value of the first 50 neighbors. It took around one hour of computation: the results for each method are in the figure below. For method 1 the best accuracy is at *K = 45*, while for method 2 is at *K = 1.* It is clear that for even values of K the accuracy decreases a lot: the reason is that a majority vote is not possible.





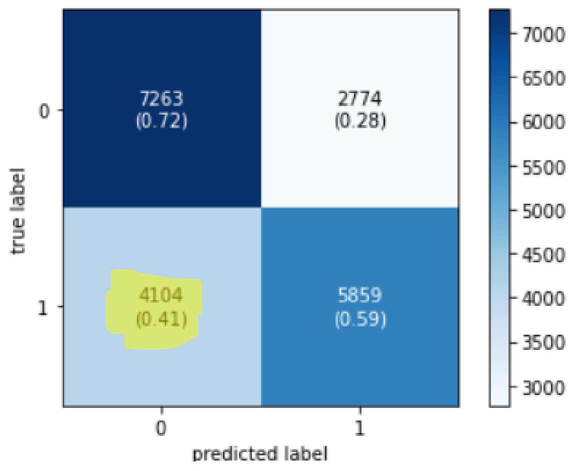
4. **Random Forest** is an algorithm based on Decision Trees: it merge methods that generate a set of decision tree, but it is not possible to control the randomness which feature is part of which tree in the forest and which data point is part of which tree. Accuracy keeps increasing as is increasing the number of trees, until it reaches a certain value. Random Forest is less sensitive to overfitting, so it is being chosen over Decision Trees.

CHAPTER 6: COMPARISON

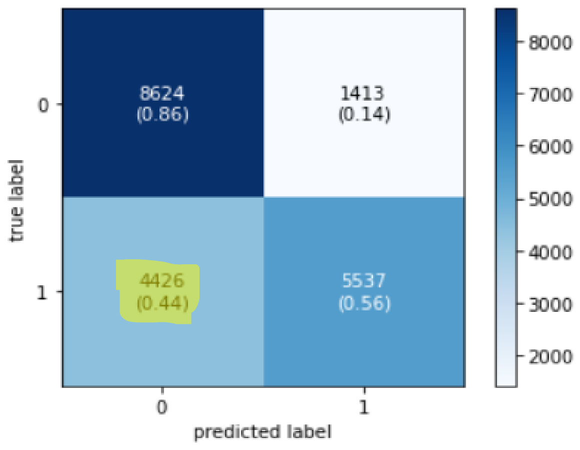
The comparison is between the 1st method and the 2nd one, using all the methods presented before. In every comparison the first images are about the 1st method.

6.1 Bernoulli NB

The computational time is not relevant in this case because it took less than one second.



```
              precision    recall  f1-score   support

           0     0.6390    0.7236    0.6787     10037
           1     0.6787    0.5881    0.6301      9963

    accuracy                         0.6561     20000
   macro avg     0.6588    0.6558    0.6544     20000
weighted avg     0.6587    0.6561    0.6545     20000
```

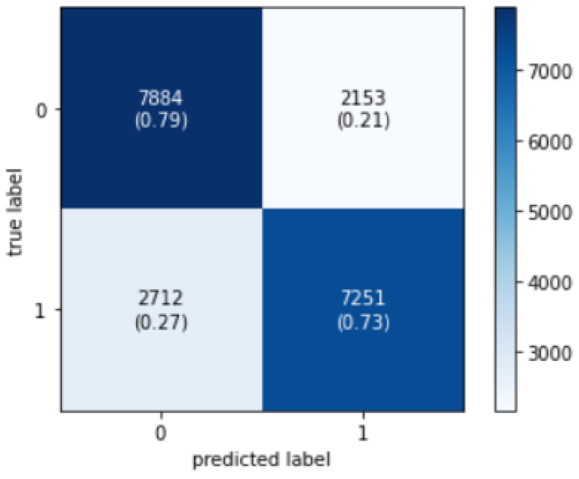


```
              precision    recall  f1-score   support

           0     0.6608    0.8592    0.7471     10037
           1     0.7967    0.5558    0.6548      9963

    accuracy                         0.7080     20000
   macro avg     0.7288    0.7075    0.7009     20000
weighted avg     0.7285    0.7080    0.7011     20000
```

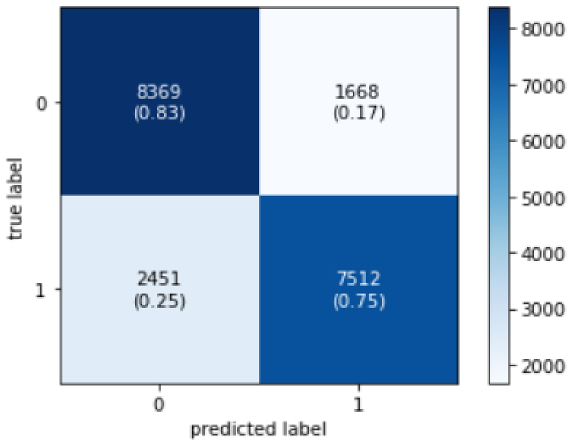
By using this method, the number of FN is high, as it is possible to understand from the two confusion matrices. The classification shows report is clear that the second method performs slightly better than the first one.

6.2 Logistic Regression

The computational time required to fit the model using the 1st method is 50 s, while for the second one it took 2.6 s.



```
              precision    recall  f1-score   support

           0     0.7441    0.7855    0.7642     10037
           1     0.7711    0.7278    0.7488      9963

    accuracy                         0.7568     20000
   macro avg     0.7576    0.7566    0.7565     20000
weighted avg     0.7575    0.7568    0.7565     20000
```

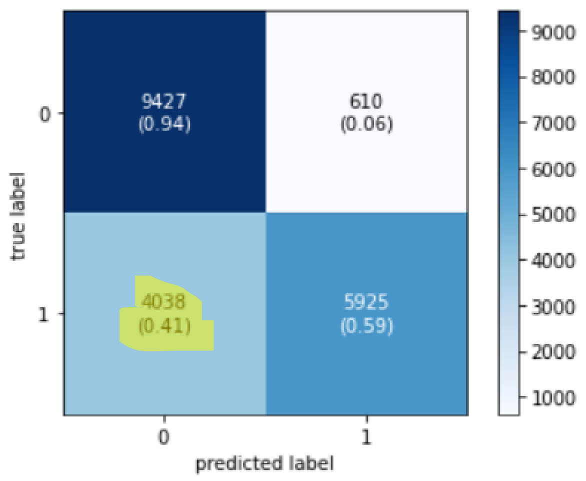


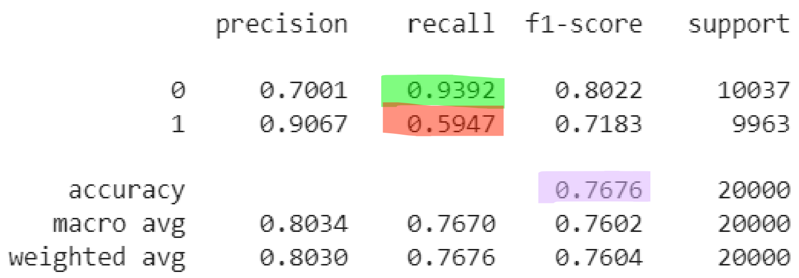
```
              precision    recall  f1-score   support

           0     0.7735    0.8338    0.8025     10037
           1     0.8183    0.7540    0.7848      9963

    accuracy                         0.7941     20000
   macro avg     0.7959    0.7939    0.7937     20000
weighted avg     0.7958    0.7941    0.7937     20000
```

The confusion matrices is pretty clear, without big issues. The classification report shows that the second method is a tiny bit better than the second. Because of that also for the Logistic Regression model is preferrable using the Tfidf Vectorization.

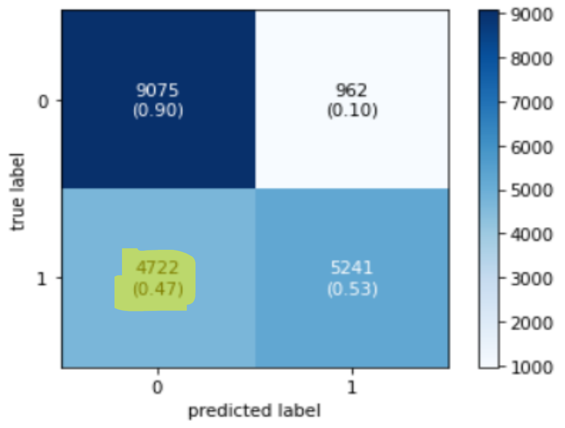
6.3 K-Nearest Neighbors

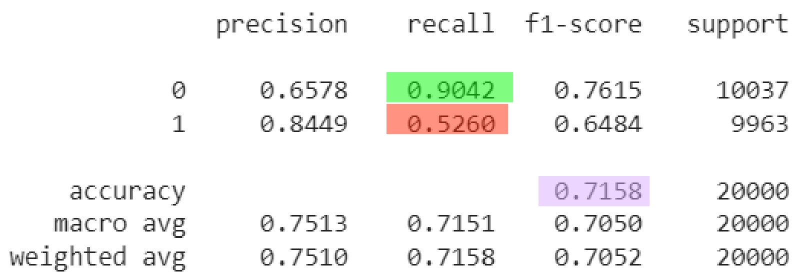
As mentioned before, firstly it is being searched the best value for K to obtain the best accuracy with K in the range(1, 50). This hyperparameters tuning setup took around 1h for each of the methods: the training is really fast, but of course the prediction (*model.predict*) takes more than 1min for each value of K.





```
              precision    recall  f1-score   support

           0     0.7001    0.9392    0.8022     10037
           1     0.9067    0.5947    0.7183      9963

    accuracy                         0.7676     20000
   macro avg     0.8034    0.7670    0.7602     20000
weighted avg     0.8030    0.7676    0.7604     20000
```



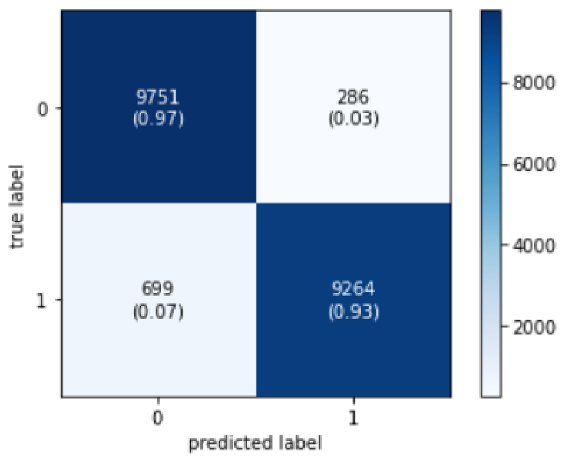


```
              precision    recall  f1-score   support

           0     0.6578    0.9042    0.7615     10037
           1     0.8449    0.5260    0.6484      9963

    accuracy                         0.7158     20000
   macro avg     0.7513    0.7151    0.7050     20000
weighted avg     0.7510    0.7158    0.7052     20000
```

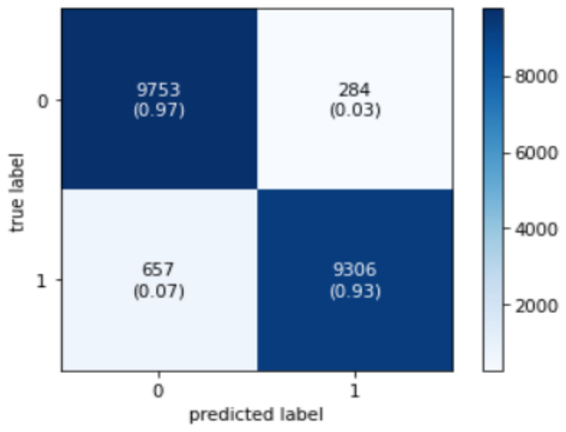
As in the NB method, there are some False Negatives. In the classification report, it is possible to notice that the first method is better than the second one. Comparing the results of these three methods, they have very similar performance metrics, but the Logistic Regression using the Tfidf vectorization is the best at the moment.

6.4 Random Forest

In this model the fitting computational time is really high for the 1st method, as reported in paragraph 6.5. With a reasonable tune of the hyperparameters it is possible to cut down the time. In particular, one possibility is given by *n_estimators* parameter that explicit the number of tree generated (by default is equal to 100). According to *TM Oshiro, PS Perez and JA Baranauskas*[1], the best number of tree in a Random Forest is between 64 and 128. Choosing a number of 64 trees takes 1h and 30m to train the model, with suboptimal accuracy with respect to 100 trees one. Another possibility is to set *max_depth* and *min_samples_leaf* parameters to control the memory usage, complexity and size of the tree, but in both cases the accuracy will be hardly impacted.



```
              precision    recall  f1-score   support

           0     0.9331    0.9715    0.9519     10037
           1     0.9701    0.9298    0.9495      9963

    accuracy                         0.9507     20000
   macro avg     0.9516    0.9507    0.9507     20000
weighted avg     0.9515    0.9507    0.9507     20000
```



```
              precision    recall  f1-score   support

           0     0.9369    0.9717    0.9540     10037
           1     0.9704    0.9341    0.9519      9963

    accuracy                         0.9529     20000
   macro avg     0.9536    0.9529    0.9529     20000
weighted avg     0.9536    0.9529    0.9529     20000
```

From the confusion matrices it is possible to observe that there is not much uncertainty: the main diagonal is full of samples, so the expected accuracy should be high. In fact, analyzing the classification report, for both method the accuracy is close to 1. Because of that, it is possible to conclude that for this problem, using both Hashing and Tfidf vectorization, Random Forest is the best method, among the four that are presented.

6.5 Computational time

In the table below it’s possible to compare the training time for each model and the accuracy

	Bernoulli NB	Logistic regression	K-Nearest Neighbors	Random Forest
Model 1	0.1132 s	50.2 s	0.0169 s (65.8 s for prediction)	8889 s (≃2.5h)
Model 2	0.0479 s	2.62 s	0.0156 s (53.4 s for prediction)	145 s
AVG Accuracy	0.6821	0.7755	0.7417	0.95

CHAPTER 7: BLIND TEST

The blind test contains 10K samples containing only *instructions* and *source_line.* There is the chance to choose among the vectorizer adopted and methods from the four that are being presented: the choice falls into the Tfidf vectorizer and the Random Forest method, because are clearly better than the other. There are five phases:

1. Load the blind test dataset;
2. Pre-process the dataset, split and fit the model chosen;
3. Once merged *instructions* and *source_line*, pre-process the blind test data-set;
4. Classify the data of blindest;
5. Write results on file *1835108.txt*.

REFERENCES

1. TM Oshiro, PS Perez and JA Baranauskas (2012). How many trees in a random forest?. 8th International Conference on Machine Learning and Data Mining in Pattern Recognition.