UCLM - Escuela Superior de Informática - Ciudad Real

Machine Learning
Lab Book

Made by:
Benjamín Cádiz de Gracia
Iván García Herrera
Dídimo Javier Negro Castellanos

Date: October, 2018.

Índice

Parte I
Milestone 1

1. Monday, 15 October 2018

1.1. Steps

- First, we have created a *cookiecutter* based directory structure.
- Then, we have revised the dataframe.
- We have deleted non numerical columns, it means, the columns that contain string values.
- We have deleted the columns which had all `NaNs`. If we don't have care enough and we try to delete rows with `NaNs` directly, we will delete the complete dataset, because there are some columns with all null values.
- We have removed duplicate rows, that is, two or more rows with exactly the same values. We have seen that in this dataset there are not duplicate rows.
- We have deleted the rows containing not a number (`NaNs`) values.
- We have tried to identify some outliers in some columns but we haven't have success.

2. Thursday, 18 October 2018

2.1. Steps

- We get the columns that ends with "MEAN" because that columns represent the average values. In that way the dataset is more representative. After that, we work on the visualization.
- We have done a `PCA` attempt but the `PCA` fails because we haven't deleted the version column and there are rows where the version number is not a float, it's a string like "x.x.x", so we eliminate the version column. The covariance is pretty high, about 0,85.
- Next, we start clustering with k-means. We analyse different groups in order to interpret the plots. We distribute the job: Ivan analyse the magnetic field, Benjamin the gyroscope and Dídimo the accelerometer.
- If we do k-means with the complete dataset, the computer is out of memory, that is the reason why we only consider the values of 5 days.
- We do k-means in a range of 2 to 50 clusters to know what is the ideal number of clusters. In order to get it, we use the *silhouette* method and we analyse which is the best coefficient. When we already know the ideal number of clusters, we plot the graphic, showing the different groups.

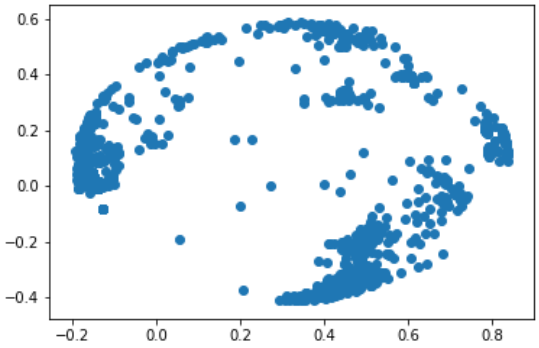


Figura 1: Principal Component Analysis

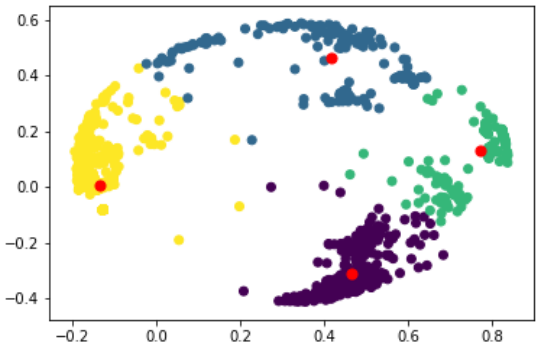


Figura 2: K-means clustering algorithm with best number of clusters

3. Friday, 19 October 2018

3.1. Steps

- We have cleaned the source code because we have seen that the implemented functionality was correct, so we have decided to organice the code in classes to avoid the repetition of code. In this way, the structure is clearer and it's easier to understand the code.

4. Tuesday, 23 October 2018

4.1. Steps

- We have studied the operation of clustering algorithm to identify outliers, which is DBSCAN
- We have based in the moodle example and we have tried to do `DBSCAN` with a symetric matrix but we didn't understand the results.
- Then we decided to use a conectivity matrix as input for a hierarchical clustering algorithm (`AgglomerativeClustering`). This conectivity matrix was obtained applying the *"k nearest*

neighbors" algorithm. This algorithm returns a matrix that indicates what k neighbors is connected an element from the matrix `AgglomerativeClustering` returns a clusters hierarchy.

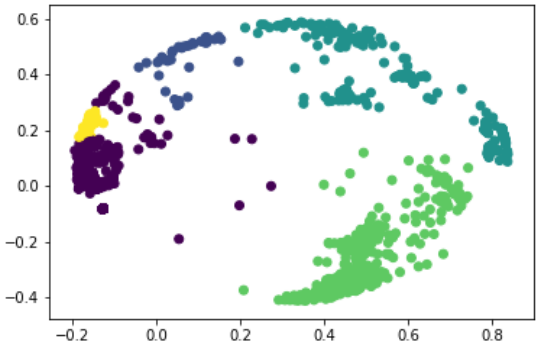


Figura 3: Agglomerative Clustering algorithm using a knn conectivity matrix

- We have applied the `DBSCAN` algorithm to know what points from the dataset are core points, what points are border points and which ones are noise. The noise points will be considered outliers.
- We have plotted the `DBSCAN` graphics to see the outliers.

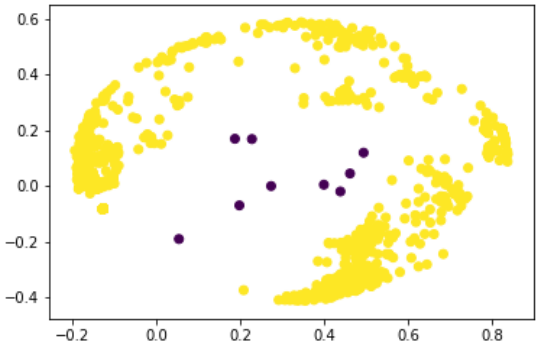


Figura 4: DBSCAN algorithm, used to identify outliers

5. Monday, 29 October 2018

5.1. Steps

- We have analysed and interpreted the K-means groups and the outliers.
- To achieve the previous point, we have grouped data by "*labels*" column and the columns with `X, Y, Z` contains the average of all data of distinct groups.
  To know what rows from the original dataset are outliers, we have added "*labels*" column to

the dataset. This column contains a "−1" value if the row is an outlier. We have filtered the rows which had a "−1" value in the "*labels*" column and we have obtained the outliers.

- He have plotted both interpreted results, the k-means groups and the outliers in a 3D graphic, using the original dataset. In this graphic (see Figure 5), we can see where are the outlier points with regard to the other points of the dataset.

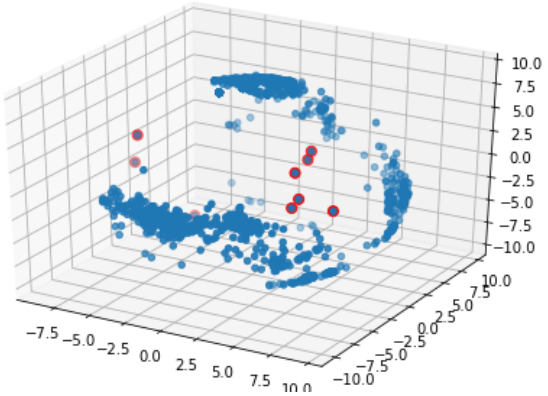


Figura 5: The outlier points are shown in red color. The blue points are all the other dataset points

6. Sunday, 4 November 2018

6.1. Interpreting results

6.1.1. Accelerometer

- In the first group we can conclude supposing that the person is stand up because the axis Y has got the maximum value.
- The second and the third group we can suppose that the person is walking because the values isn't so clearly.
- The fourth group has got values near to 0 and because that we can suppose the mobile phone is in repose.

Index	lerometerStat_x_M	lerometerStat_y_M	lerometerStat_z_M
0	0.248045	-9.46863	-0.520377
1	6.38539	4.4016	-2.16908
2	6.07473	-3.87995	-6.87392
3	-0.343393	-0.158427	8.7404

6.1.2. Gyroscope

- In the first and third group with the values we can't see anything clear. We can suppose that the person is walking.

- In the second group we can see how the mobile is in repose.

Index	oscopeStat_x_ME	oscopeStat_y_ME	oscopeStat_z_ME
0	0.0848001	-0.350585	-0.0511788
1	-0.00948414	-0.005377	-0.00776391
2	-0.084057	0.349342	-0.0117479

6.1.3. Outliers

In the accelerometer outliers hasn't relationship with each other but we can see that this values are near with other outliers, so it's possible that the person had done something stranger.

On the other hand, in the gyroscope outliers is the same case that accelerometer, but we can see the all outliers values are near to 0. It's all we can say because we haven't got more data

7. Repository Link

`https://github.com/ivangarrera/MachineLearning`

8. Video Link

`https://www.youtube.com/watch?v=iqD1iOzJJwU`

Parte II

Milestone 2

9. Monday, 26 November 2018

9.1. Steps

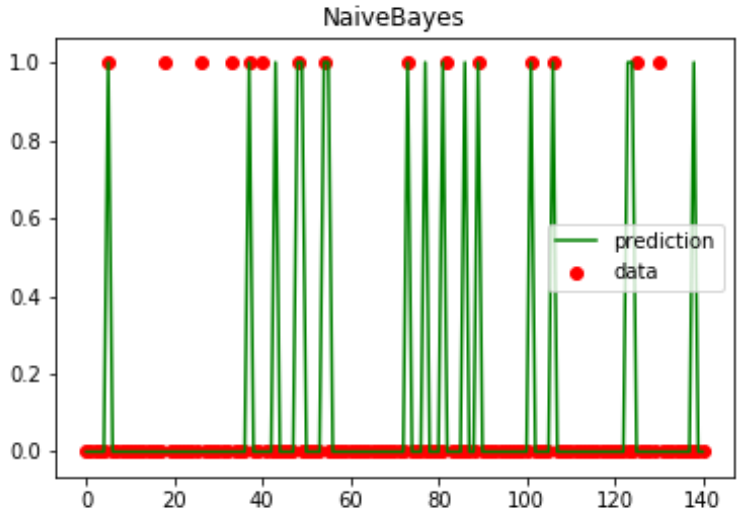
We choose the T4 dataset because we think are more relevant respect T1, T2 and T3. We tried to merge Moriarty and T4 dataset but we had a problem since the *UUID* didn't match because it's a value in milliseconds and it's very difficult to be exactly the same. We did the merge manually. We are going through the *moriarty dataset* and if the UUID is in an interval of 5000 we put the same UUID since we consider that it is in the same moment of time. Finally, with *pandas* we made the merge with the uuid key.

10. Tuesday, 27 November 2018

10.1. Steps

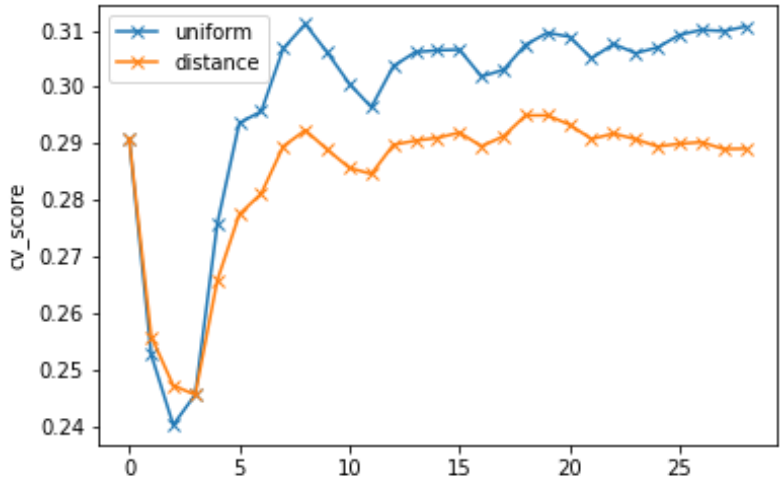
- We tried to merge the dataset with `merge_asof` but we had the problem that the data didn't associate the correct form.

  After that, we applied *Naive Bayer* to make the prediction and see the results with that prediction. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.



  The line represents the prediction of Naive Bayes and the points the real data. When the point coincides with the line, the algorithm is successful, otherwise it is considered a failure

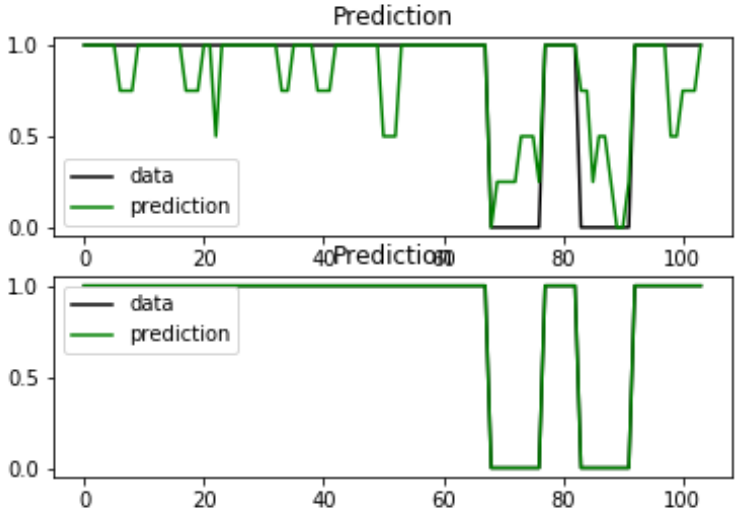
- Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.



- **Uniform weights**: each point in the local neighborhood contributes uniformly to the regression of a query point.
- **Distance weights**: weight points such that nearby points contribute more to the regression than faraway points.

The intersection of the two measures gives us the optimum number of neighbors.

- From the optimal number of neighbors we did the regression.



11. Thursday, 29 November 2018

11.1. Steps

We have cleaned the source code because we have seen that the implemented functionality was correct, so we have decided to organice the code in classes to avoid the repetition of code. In this way, the structure is clearer and it's easier to understand the code.

Parte III
Milestone 3

12. Wednesday, 05 December 2018

12.1. Steps

We have planned a meeting with the teacher to revise the practice. The basis of the problem was correct but we had faced the problem badly. We were making the prediction in base to moriarty dataset and predict which were malign and which were benign. For the milestone 3, We have to merge the Moriarty data with another dataset of Sherlock, that in our case is T4, and then we have to predict when the attacks will occur.

13. Monday, 10 December 2018

13.1. Steps

- We needed to add a column whose indicates when the attacks occur. For it, we made a query with *pandas* and it return the values between the indicated interval. If the Moriarty timestamp is near to the Sherlock, we add 1 value to the column added about attacks in Sherlock.

```
1  def MergeDataByUUID(self, characteristics):
2          self.data_set = self.data_set.loc[:, self.data_set.columns.str.
               contains('|'.join(characteristics))]
3
4          for index in range(len(self.data_set_moriarty["UUID"])):
5                  self.merged = self.data_set.query("UUID <= " + str(self.
                       data_set_moriarty["UUID"][index]) +" <= UUID + 5000")
6                  if not self.merged.empty:
7                          self.data_set_moriarty["UUID"][index] = self.merged["
                               UUID"].values[0]
8
9          self.merged = pd.merge(self.data_set, self.data_set_moriarty, on="UUID
               ")
```

- We had 150.000 rows in T4 dataset so we need to reduce that. For that, we thought that approximately 300 values were enough, so we chose 300 random rows. We tried that the values were distributed throughout the dataset and not only on the beginning, the end or all

together. Later, we added all rows with Moriarty attacks because they are much less than the total rows of dataset and It was very improbable we haven't caught them with that random algorithm. Finally, we sorted the reduce dataset by timestamp.

```
def CreateSupervisedDataset(self, number_of_non_attacks):
        mydataset = pd.DataFrame(data=None, columns=self.data_set.columns)

        # Select non-attacks and include them into the dataset
        for i in range(300):
                index = int(i * len(self.data_set) / number_of_non_attacks)
                frames = [mydataset, self.data_set.iloc[[index]]]
                mydataset = pd.concat(frames)

        # Select attacks and include them into the dataset
        for i in range(len(self.data_set)):
                if self.data_set["SessionType"][i] == 1:
                        frames = [mydataset, self.data_set.iloc[[i]]]
                        mydataset = pd.concat(frames)
        return mydataset
```

14. Tuesday, 11 December 2018

14.1. Steps

- We implemented *KNearestNeighbors* algorithm. We did the classification with a range of neighbors from 1 to 30 and we saved the neighbor with the best score. Then we did *KNN* with the best neighbor.

- We implemented *Random Forest Classifier*. Random forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.Random decision forests correct for decision trees'habit of overfitting to their training set.

15. Wednesday, 12 December 2018

15.1. Steps

- We were doing the prediction with *zero_one_loss* metric that it returns the fraction of misclassifications. The results weren't so good and we didn't get a clear interpretation of them.

- After trying different metrics we chose *hamming_loss* metric that it returns the fraction of labels that are incorrectly predicted. We obtain better result so we realized *KNN* with this metric.

16. Thursday, 13 December 2018

16.1. Steps

After trying different methods of classification and prediction we proceed to analyze them and see which obtain a best prediction.

- ***Naive Bayes*:**

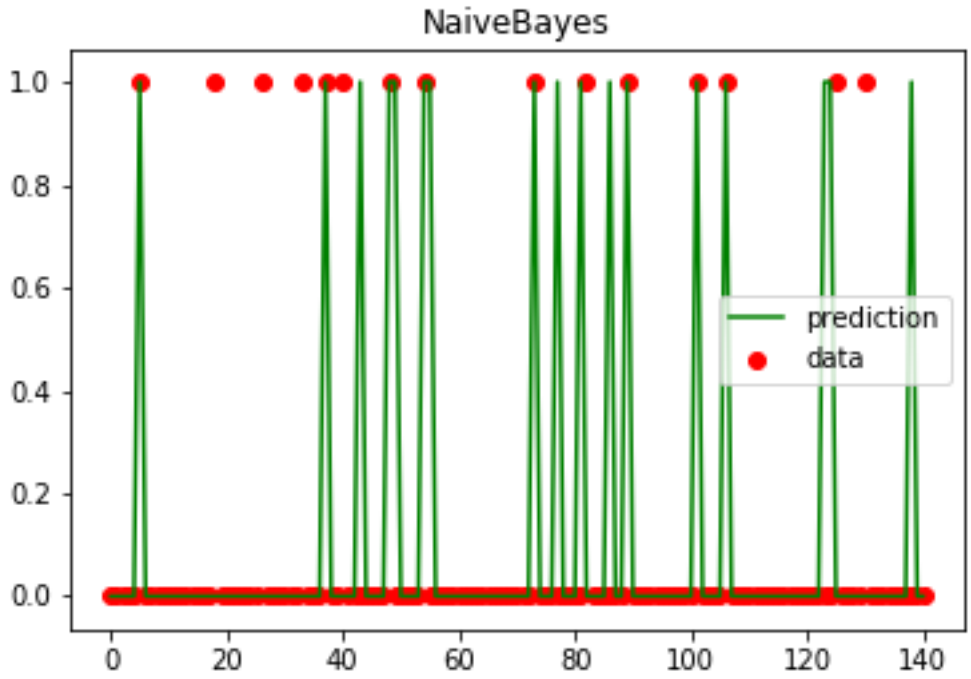


Figura 6: Naive Bayes prediction

This is the algorithm which predicts more attacks. In addition, there are very few false positives. It has a error measure of 0.12 approximately, so it means that 12 values of 100 are incorrectly predicted.

- ***KNN:***

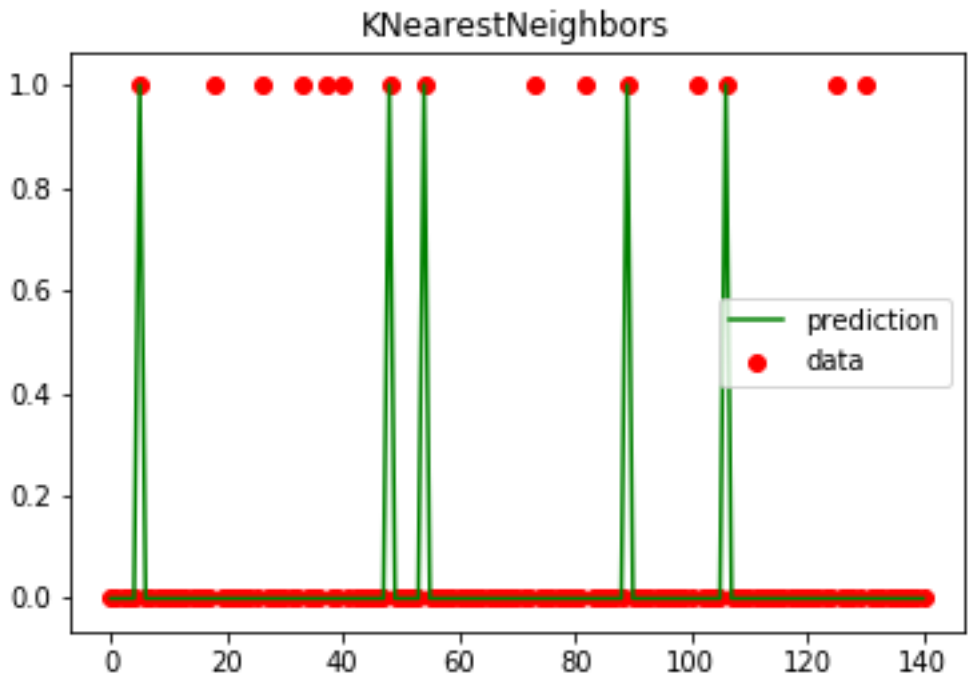


Figura 7: KNearest Neighbors prediction

We obtain an insignificant error measure such as 0.07. It seems is a perfect algorithm because it always hits the prediction, but we can see in the graph that it only predicts when there are no attacks. In spite of that, when it predicts an attack it is usually correct.

- ***Random Forest:***

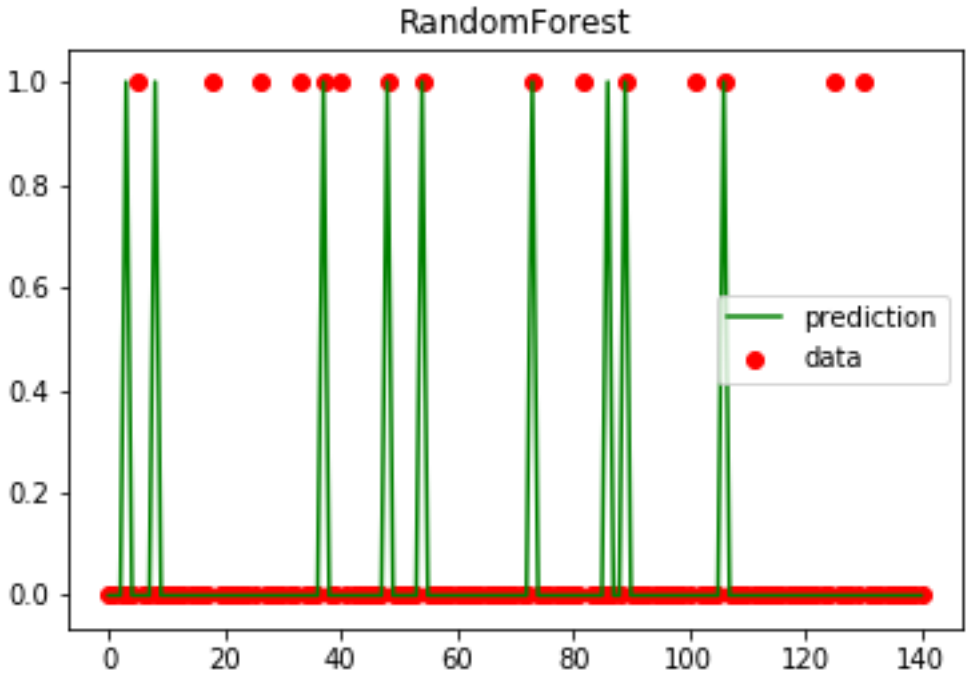


Figura 8: Random Forest prediction

As in the algorithm *KNN* it has a very low error. In this one, more false positives are observed and it not predict enough attacks.

The best algorithm to predict attacks is *Naive Bayes* since it predicts more attacks and although it has a greater error than the other two it is still very low. With a better training model we could predict more attacks, but in this case the data with attacks are significantly lower than those that don't receive attacks.