ALGORITHMICS FOR DATA MINING

ADM - ASSIGNMENT 2

April 29, 2018

Fabio Di Francesco
David Sarda

Contents

1 INTRODUCTION

We are big python fans so we decided to try one of the libraries for machine learning that are available, *sklearn*. To do so we chose a dataset that looked interesting, it's about homicide reports on the United States between 1980 and 2014 (the link to the database is https://www.kaggle.com/murderaccountability/homicide-reports/data). The main variables are as follows:

City/State/Year/Month: Time and place of the incident.
CrimeType: The type of crime commited, in this database it can be either a murder, or a manslaughter by negligence.
Crime solved: It states whether the crime was solved or not (by 2016).
Victim_sex/Victim_age/Victim_race: Characteristics of the victim.
Perpetrator_sex/Perpetrator_age/Perpetrator_race: Characteristics of the main perpetrator.
Relationship: The relationship between the victim and the perpretator.
Weapon: The weapon used for the crime.
Additional_Victim_Count: The number of victims killed by the perpetrator in addition to this one.
Additional_Perpetrator_Count: The number of perpetrators that collaborated with the main perpetrators to execute the crime.
Record_Agency: The agency that was responsible for investigating the case.

2 PREPROCESSING

This database was already in a good state from the start, there weren't any missing values, and the values meaning were pretty clear in general. So the main point we needed to tackle was that more or less half of the variables were categorical (and some, like weapon, had up to 15 categories), so we converted all of them to numerical. In some of these columns there was a special value called unknown that meant that the investigation didn't have information on that variable, so we translated this unknown values to the highest numerical value possible. We also got rid of some of the columns that weren't clear or didn't have helpful values because they were only to specify technical data, like the record ID, the agency code and the record source.

3 DATA MINING ALGORITHMS

3.1 Naive Bayes

The first thing we wanted to see was if we could predict if a crime would be likely to be solved or not. For this purpose, we tried a naive bayes classifier, as it seemed a good fit for predicting a binary variable, and also because it's simple to implement, and we could introduce ourselves to the sklearn library. We started with partitioning to get the first results quickly, and they were pretty surprising, as can be seen in figure 1.

```
partitioning bayes crime solved confusion matrix:
[[ 56026   1237]
 [    41 134232]]
Accuracy: 0.9933
```

Figure 1: Naive Bayes confusion matrix using partitioning

Even if we tried to change the training and test set, the results were almost identical, so we knew that this kind of accuracy was likely due to some error in the process. Just to be sure, we tried to see if the results changed significantly with cross validation, as can be seen in figure 2, but to no avail.

```
cross validation bayes crime solved confusion matrix:
[[16438   632]
 [    0 46776]]
[[16214   641]
 [    0 46991]]
[[17174   741]
 [    0 45931]]
[[19546   387]
 [    0 43912]]
[[20450   206]
 [    0 43189]]
[[19815   197]
 [    0 43833]]
[[19041   190]
 [   23 44591]]
[[20010   342]
 [   24 43469]]
[[19218   331]
 [   43 44253]]
[[18317   391]
 [   57 45080]]
Accuracy: 0.9934 (+/- 0.0058)
```

Figure 2: Naive Bayes 10-CV confusion matrices-

3.2 Decision Tree

After the results given by the Naive Bayes predictor, we were really interested to see if and what variables were causing this anomaly. We thought a decision tree could give us that

information, and also it seemed like it would be a good fit for the problem. As before, we configured the classifier to predict if a crime was solved or not. The results of the cross validation were similar to Naive Bayes, as seen in figure 3.

```
[[17070     0]
 [    0 46776]]
[[16855     0]
 [    1 46990]]
[[17915     0]
 [    0 45931]]
[[19933     0]
 [    0 43912]]
[[20656     0]
 [    0 43189]]
[[19964    48]
 [    0 43833]]
[[19214    17]
 [   77 44537]]
[[20289    63]
 [   24 43469]]
[[19535    14]
 [   43 44253]]
[[18599   109]
 [   52 45085]]
Accuracy: 0.9993 (+/- 0.0008)
```

Figure 3: First Decision tree 10-CV confusion matrices

So afterwards we proceeded to look at the construction of our predictor. The variables that were used in the high levels o the tree proved very helpful to determine the cause of such high accurate scores, as we can see in figure 4.

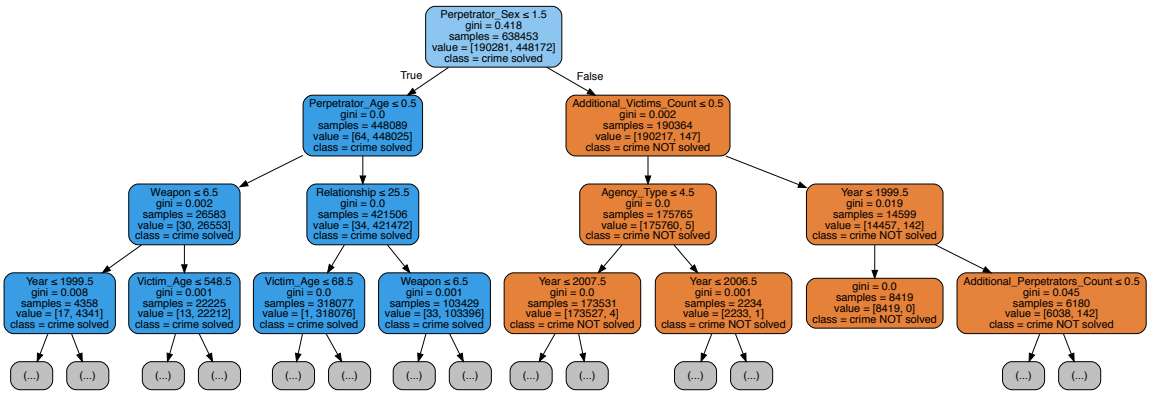

Figure 4: First Decision tree

As we can see, the main way to distinguish between a crime solved and a crime not solved is to see what is the sex of the perpetrator. If the sex is unknown(value 2), then the crime is almost always not solved! But this is obvious, so we don't want to consider this variable to predict if a crime is solved or not. In the same fashion, the perpetrator age can classify if a crime has been solved too accurately. We then got rid of all variables that describes the

perpetrator and his/her relationship to the victim, and proceeded to classify the tree again. The results were much different this time, as seen in figures 5 and 6.

```
[[ 5658 11412]
 [11641 35135]]
[[ 5647 11208]
 [11504 35487]]
[[ 6330 11585]
 [12745 33186]]
[[ 7506 12427]
 [12891 31021]]
[[ 7976 12680]
 [13434 29755]]
[[ 8005 12007]
 [14244 29589]]
[[ 6913 12318]
 [13143 31471]]
[[ 6891 13461]
 [11689 31804]]
[[ 7645 11904]
 [13145 31151]]
[[ 9077  9631]
 [15820 29317]]
Accuracy: 0.6104 (+/- 0.0180)
```

Figure 5: Second Decision tree 10-CV confusion matrices

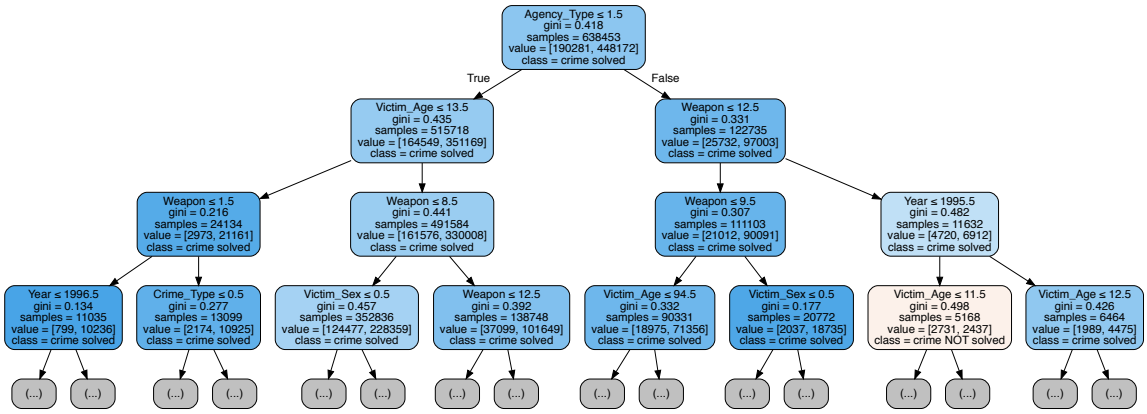

Figure 6: Second Decision tree

As we can see, the classification is much worse, but seems like a legit one. We can observe the main ways to know beforehand if a crime is hard to solve:

1. The county or the municipal police are the police department that investigates the case. This is logical, as a smaller police department will have less means to catch the culprit (and also specialized departments are only called when they know they will perform better than more common means.

2. The weapon is not known. This is fairly straightforward, as knowing the weapon used in the crime leads to key clues to solve it.

3. The year of the crime is older or equal to 1995. This is also logical, as technology evolves very rapidly nowadays, and new crucial techniques for investigation become available in a matter of years.

So, even if the classification has only an accuracy of approximately 60 percent, we managed to understand the key characteristics of tackling the problem of solving a crime which is what we were looking for. Even then, we wanted to try to implement a roc curve visualization of the classifier. The results can be seen in figure 7

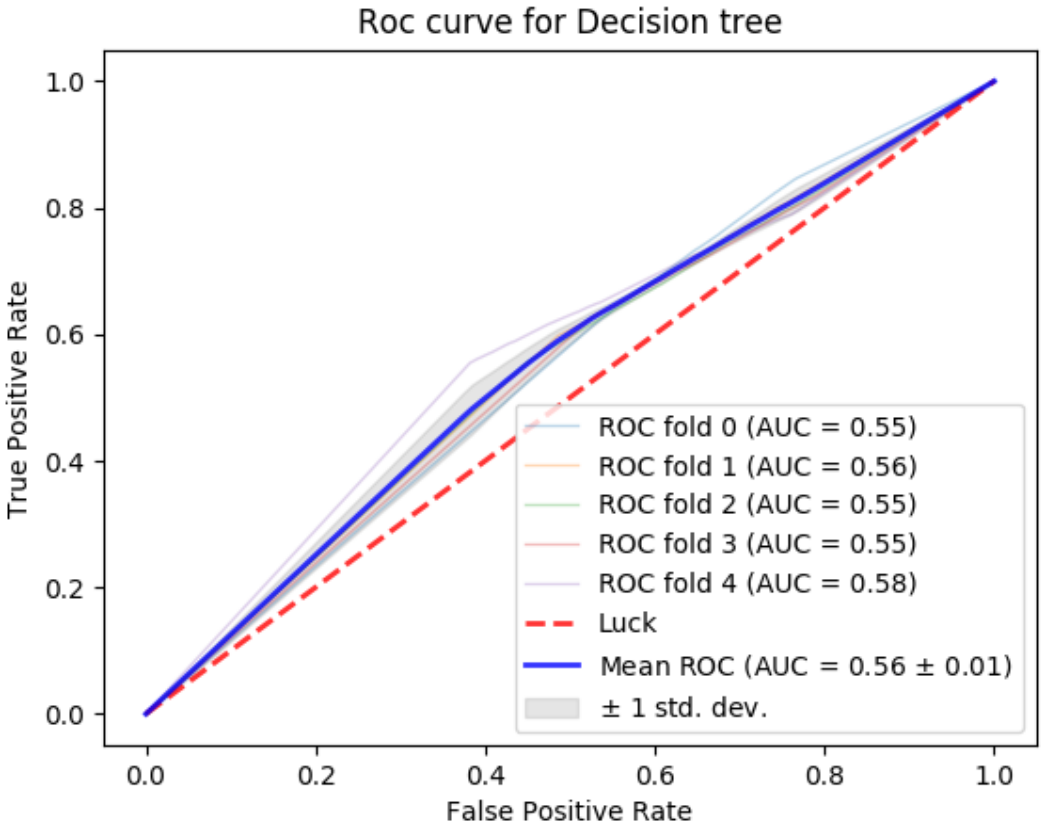

Figure 7: 5-CV ROC curve using decision tree

We can see that the area of the rock curves is pretty low, meaning that this is not a great classifier. Still, at least it seems to provide consistent results across all the folds of cross validation.

3.3 K-Nearest Neighbours (KNN)

Another algorithm that we wanted to test because is very common and easy to use was k Nearest Neighbour. As the other algorithm, the usage is very simple as it mainly requires the number of neighbours and the train and test dataset to work.

3.4 Predictor Application

As a first test, to makes things funny, we designed a simple input window where to put the city, the gender and the age. The algorithm predicts what type of weapon a person with these inputs can die from.

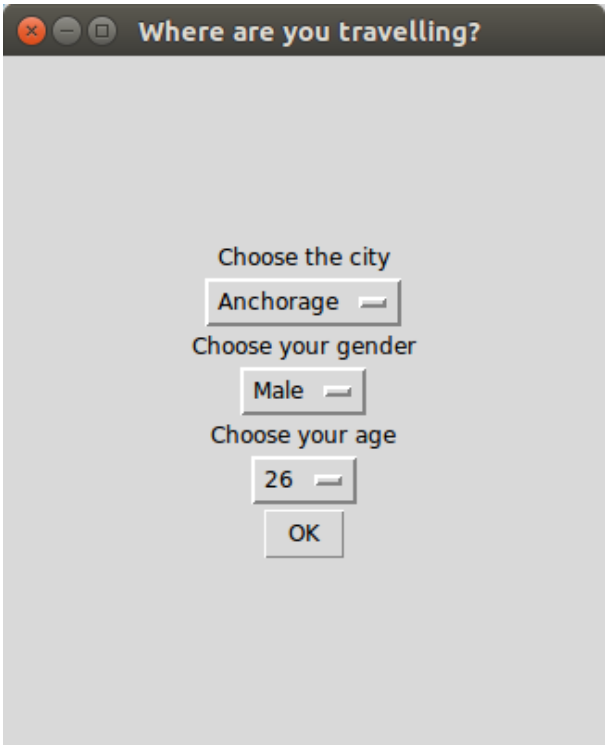

Figure 8: The input window

To do this, the algorithm was trained with the data from the city, the gender and age of the victim and the weapon used. However, testing the accuracy with a 10 cross validation shows that the algorithms is not very precise, even changing the number of neighbours from 5 to 30, it reaches a maximum of 0.5%, of course with higher numbers of k. To run this test we had to limit the number of cities to be chosen as they could not be displayed in the UI. The result that we have from this test is that the handgun is the most common weapon used (predictably in the US there are many handguns owners), but also that some other city have

```
Anchorage26Male
The victim will die with a: Knife

Albany50Female
The victim will die with a: Handgun

Apache5Female
The victim will die with a: Blunt Object
```

Figure 9: Test results

crimes committed with white weapon, like in the example showed in figure 8. Furthermore choosing a very young age, returns a blunt objects as weapon, as rarely young children are shot, except for accidents.
Then we also test the algorithm to predict the age and gender of the perpetrator of the crime to know in advance who is the killer. This time we can observe that the accuracy of the algorithm to predict the gender is much higher than the previous case, even leaving the default value of k equals to 5, and also better than the prediction of the age.

```
Perpetrator sex confusion matrix:
[[  3460  11038    104]
 [  3201 113349   3033]
 [    29    387  56935]]
```

Figure 10: Confusion Matrix for the gender of the perpetrator

```
knn Perpetrator sex accuracy: 0.9034 (+/- 0.0204)
knn, k = 5 Perpetrator age accuracy: 0.3381 (+/- 0.0540)
knn, k = 8 Perpetrator age accuracy: 0.3405 (+/- 0.0562)
knn, k = 11 Perpetrator age accuracy: 0.3428 (+/- 0.0569)
knn, k = 14 Perpetrator age accuracy: 0.3443 (+/- 0.0567)
knn, k = 17 Perpetrator age accuracy: 0.3451 (+/- 0.0574)
knn, k = 20 Perpetrator age accuracy: 0.3457 (+/- 0.0578)
knn, k = 23 Perpetrator age accuracy: 0.3462 (+/- 0.0582)
```

Figure 11: Accuracy of knn test

This results were expected, as the algorithm can classify easily an element when the classes are few, like the gender, but struggles with more classes, like the weapon and the age. The performances of the algorithm were quite good. Of course testing the algorithm with different number of neighbours took a lot of time, but on average the time necessary was coherent with the dimension of the database.

4 FINAL THOUGHTS

After trying out this library we have reached some mixed feelings about it:

+ It provides a very deep way of implementing machine learning, that is to say that the functions included on this library are very specific in their function, and you are free to implement your own way of doing stuff.

+ Being in python mean that it may be one of the best ways to intertwine both algorithm programming and machine learning easily.

+ The number of available algorithms for this library is really extensive.

- It's not that clear what is happening on each step, even when using python IDEs or printing outputs, you hardly get a clear idea of the process that your data goes through, or what's the state of it at each step.

- The documentation is confusing at times, and relatively not that many people uses this library.

- The classes included could use more functions at times, and this functions would also improve if they could adapt to different inputs and outputs (for example, needing that all variables to be numerical, and not doing the conversion internally).

In conclusion, we think we would use this library only when we already had a clear idea of the data and a specific objective in mind. Also it can be useful if you need to implement advanced algorithms alongside your machine learning code.

Nevertheless, to get a first idea of the data, and in most cases in general, Knime seems to be much more useful and fast, or if we needed something more specific R should provide a more clear way to implement machine learning.

In respect to the database itself, it seemed like a very fun database to play with, having a lot of different ways you can test different techniques. Also the conclusions we reached on each of the sections were very interesting.