Car Crash Project

[1]Aliyev Gurban,[2]Barandoni Simone,[3]Fanelli Gioele, and[4]Parshina Kseniia

University of Pisa, Italy

Abstract. The research is aimed at car crash prediction classification task. It takes into account the previous data collected and focuses on Recall evaluation. Three different areas of individual mobility networks are observed in order to train a machine learning algorithm

Keywords: Car Crash · Recall · Data Mining · Prediction

1 Project Understanding

1.1 The Dataset

It is a dataset of around 400 features describing the mobility of around 5,000 private vehicles traveling during four months. The features are extracted from GPS tracks that are produced by on-board GPS devices. The GPS device automatically turns on when the vehicles start, and the global trajectory of a vehicle is formed by the sequence of GPS points that the device transmits each 30 seconds to the server via a GPRS connection. Moreover, records of car crashes are available, describing the vehicles involved.

Can we predict future crashes? The main Goal of the project is to train a model able to detect future crashes. Furthermore, the target variable to predict is whether the private vehicle will have a car crash in the month following the months the mobility data refers to. For this problem, it is also important to explain (if possible) why car crashes happen.

2 Data Understanding

This research is aimed at car crash prediction using individual mobility networks, created by measuring different parameters regarding car travels in three different places, London, Rome and Tuscany along four months. The binary result of classification will be valuable for identifying the risk of a car accident, but mainly focused on Recall.

2.1 Data Semantics

As an initial data set we have 21 train and 21 test sets, 7 of each collected in London, Rome and Tuscany. Tuscany datasets are composed by 429 attributes, while London's and Rome's data sets have, respectively, 2 and 6 features less than Tuscany. There is also a difference in the number of records present in the datasets. On average, Tuscany datasets present more records than the others (the average is 5834), Rome datasets have the smallest amount of records (3706 on average), and ones of London are close to ones of Tuscany (5590 on average). There is a vast number of features because a lot of different statistics and data about car travels have been recorded: the acceleration, the speed, the trajectory, the number of events that occurred during the travel, the most frequently visited places, and all is also registered for four parts of the day, morning, afternoon, evening and night. Among the total of features, we identified the uid column, which stands for user id and is useful to understand the index of the datasets, and the crash column, which is the target variable. Features related to the mobility behavior of the private vehicles could be divided into groups: single trajectories of the vehicles (count, length, durations, time of day, speed, etc.); specific driving events (accelerations, breaks, cornering, start/stop); recurrent mobility patterns (a graph abstraction of the overall mobility of the individual based on locations (nodes) and movements (edges)). We have focused our attention on the target variable, crash, a feature containing only binary values, 0 and 1, meaning, respectively, that the car crash didn't happen and that the car crash happened. All the attributes have a scale type called numeric, and only variable crash is a categorical one.

2.2 Distribution of the variables and statistics

We executed statistical analysis on all the training datasets available, and we compared the results, especially to obtain an overview on which could be the differences not only among the different parts of a day, but also among three areas taken into account. We obtained interesting results concerning the target variable crash. We extracted statistics about more meaningful features, like the average speed, acceleration and trajectory. Taking average speed case as an example (Fig.1), we plotted the distribution of values to understand how to discretize the records, as it is possible to observe in the following plot (first boxplot: average speed in London, second boxplot: in Tuscany, third boxplot: Rome).

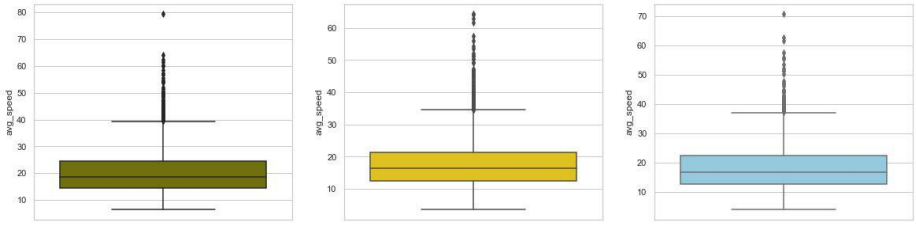

Fig. 1.

We understood that the distribution of this column is similar in three different areas, this allowed us to discretize the records in the same way for each area.

The records have been divided into 4 groups of the same size, from low to high average speed. After the discretization, we plotted the different bins showing the percentages of yes and no values of crash column (Fig. 2), and we noticed, for example, that the records presenting a bigger average speed, have yes to crash with more probability, as it is possible to see in the following graph. (low average speed has a percentage of crashes equal to 4, *medium* and *normal* have it equal to 5 and 6, high has a percentage of 6 %).

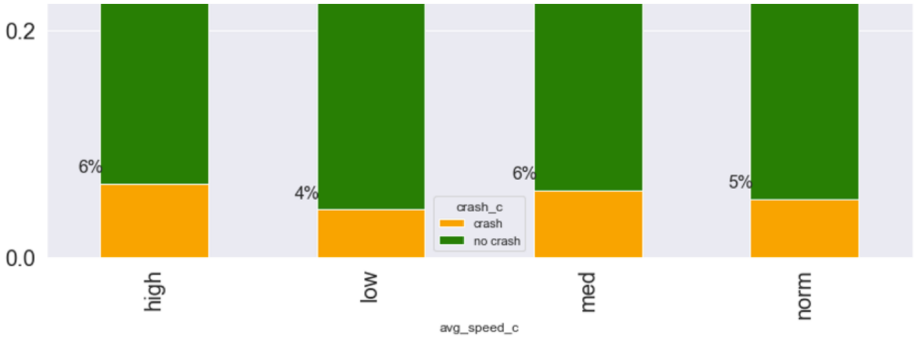

Fig. 2. Velocity and crash percentage

Analyzing the crash variable in particular, we discovered a big difference in the number of values: in all the datasets, only a number between 1 % and 3 % of the records have yes (1) as a value for the crash. This means that only an extremely small percentage of the travels recorded ended with an accident; we decided to face this problem of unbalanced data at the end of the Data Understanding part and before Classification.

Regarding the differences between areas, in general, in all the datasets, the percentages of records where the crash happened (value yes in the attribute), were always very low, but we noticed a relevant difference between London and two Italian areas (Fig. 3).

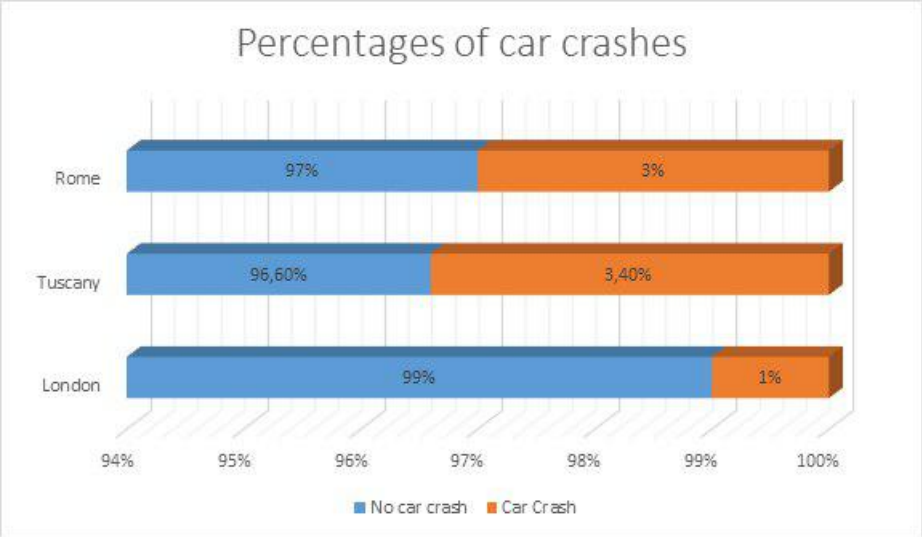

Fig. 3. The three Cities

As it is possible to observe from the bar plot, in Tuscany and Rome the percentages of travels ended in a crash is around three times the one registered in London.

2.3 Assessing data quality (missing values, outliers)

Exploring all the datasets, we counted the number of missing values present in each feature. We decided to remove the columns containing a total of missing values which was bigger than the 1 % of the records. In this way, we removed around 6-7 % of the total features, in each dataset. The remaining missing values in the dataset have been solved by replacing them with the mean of the column where they appeared. Considering that we applied this operation to the columns with less than 1 % of the missing values only, the increase of the number of values close to the column's average was too small value to relevantly change the shape of the feature. Studying the presence of outliers, we identified the feature uid that, as already stated, represents the identifier of each car present in each dataset.

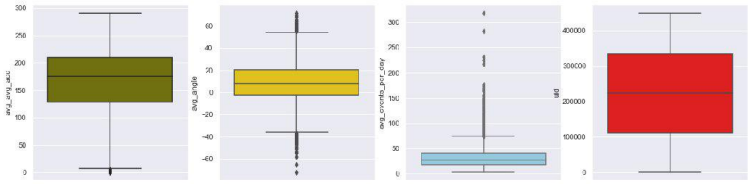

Fig. 4. UID Feature

As it is possible to observe in the figure above, the boxplot of uid (Fig. 4), the last one in red, is the only one without outliers. In fact, all the values present are unique identifiers.

2.4 Variables transformation

Considering the fact that all our attributes have numeric variables, transformation could be applied. In this way, we used the discretized value to produce graphs and observe distributions and statistics, also related to car crash percentages. For example, in the following figure, it is possible to observe the percentages of car crashes in the records organized in three groups regarding the average of kilometers done: the probability of a crash increases a little bit in the column *high* (Fig. 5).

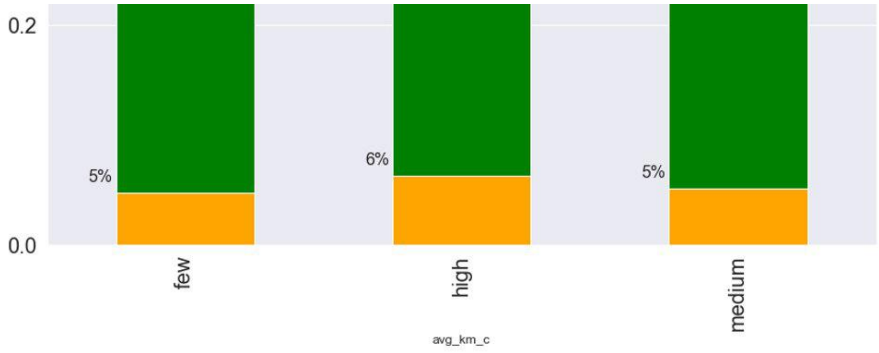

Fig. 5. Dependencies between average km and crashes

2.5 Features Engineering

We read several articles on car crash prediction with machine learning to get possible ideas about feature construction. These articles were especially useful in

issues of integrating features with external sources and invention of new features. The main article referred is written by Daniel Wilson, who constructed a model to predict car crashes. We decided to reduce the amount of columns to face the classification task in a more efficient way. Firstly, we removed from Tuscany and London the columns that were not present in Rome, to have the same columns in all the dataset. Then, we realized a correlation matrix among the attributes to observe which are the redundant features, so that we could remove some of them to obtain a more clear dataset (Fig. 6). In this report, we provided a smaller version of the full matrix because of the existence of a high number of features which created a huge matrix. The lighter the area in the figure, the higher correlation is between the values. We can observe from the matrix in the figure that average speed and average minimal speed per day have the highest correlation; consequently, we will probably eliminate one of them to reduce the dataset. The most uncorrelated attributes are, for example, crash and distance made by morning travel in km (equals 0.007197). They could be too insignificant to make a classification, but we may use those features in feature creation step.

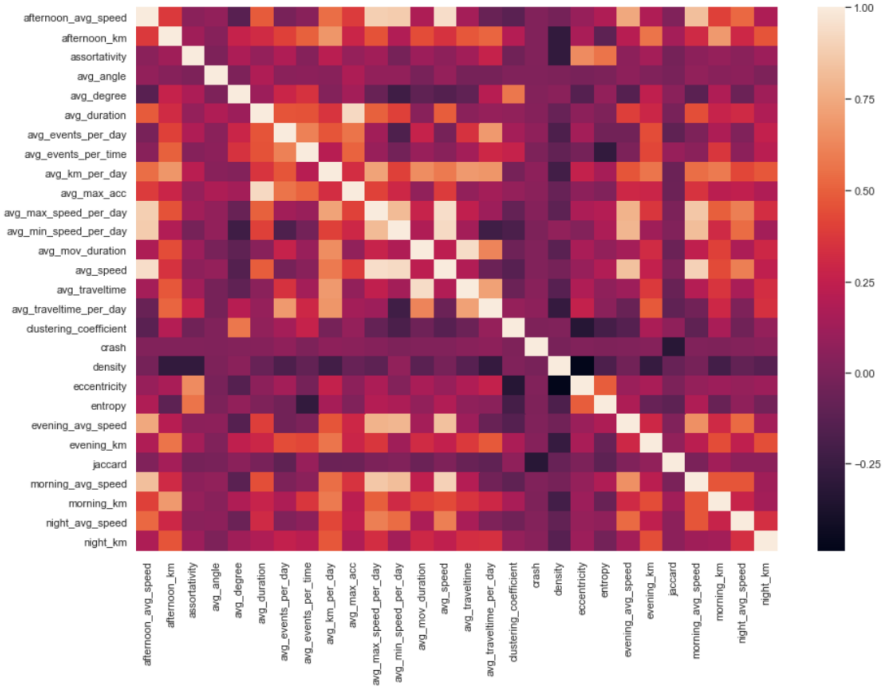

Fig. 6. Weighted Correlation Matrix

To eliminate redundant features, we decided to drop each column which had a correlation higher than 0.7 with at least another one. In this way, we reduced the

amount of columns in each dataset from 423 to 144, without losing attributes which could be relevant for the classification. Concerning integration with external sources, Wilson used not only static but also dynamic variables, which had more importance in prediction. The main features were solar position and temperature, both of which are dynamic features. Also, it would have been interesting to derive interesting variables such as sinuosity, which is is the ratio between the path length and the shortest distance between the endpoints. It was not possible to use such features as our data only includes static features aggregated in 3-month period. When it comes to static features, we had all variables derived from existing features (such as average values derived from total values).

3 Classification

3.1 Preprocessing

After working to reduce the size of the datasets and to solve the missing values, we addressed the classification task, trying many different approaches with the aim of predicting car accidents. In this task, we decided to concentrate our attention to improve the accuracy of crash prediction and its Recall. The meaning of this choice is the fact that the purpose of the model is to prevent people to crash during their next travels looking at their individual mobility network, so it is important to understand which are the feature values that the crashed training records have, to find similar ones in the test sets. The importance of the Recall is due to the fact that in a real situation it would be correct to stop a person whose travel has been predicted as a probable crash, even if the prediction was wrong. Instead, failing the opposite prediction (consider a travel safe, while then it ends in a crash) would be much worse.

3.2 Clustering

We tried to improve our dataset adding information about clustering. We executed a hierarchical clustering to observe the distribution of records in groups; and later, we created a new column recording the cluster label of each record. Our purpose was to check if this generated column could be useful during classification.

3.3 Oversampling and Undersampling

Due to the huge data unbalance, it has been necessary to oversample the minority class. Considering that we have seven datasets for each area, each taking into account three months, we decided to create a unique dataset for each group (Tuscany, Rome, London), losing the temporal perspective, built in the following way: we took all the records of seven datasets with *yes* in the crash column; considering their number, we randomly took from the same datasets a bigger number of records with *no* to crash to maintain the percentages of around 20/100

of records with yes and 80/100 of records with no. There are three final datasets, one for each city, and have around the same number of records as the initial ones, but with a good percentage of crashes inside (~ 20).

Furthermore, we created combined datasets for couples of areas and for all the areas together: London and Rome together, Tuscany and Rome together, Tuscany and London together, and the complete dataset with three areas together, composed by around 2500 crashed records and 12000 not crashed records.

3.4 First Tests and final features reduction

The first approach tried has been done using a Decision Tree classification. Each training dataset has been used to produce a model, which has been created trying to avoid overfitting (setting a limit to the MaxDepth of the tree). We noticed low results in classifying the test sets. We executed many tests tuning the features used in the model, and we found a good compromise deleting a larger number of columns. In particular, considering that for many features there is the corresponding average, total and standard deviation, we decided to maintain only the average for all measures and the total for the counts of events. Training models using these columns, the classification results improved. Also, we noticed how the clustering column didn't appear in the important features used by the decision tree. So, we decided to drop that column as well. For example, the following table shows the improvement between the test using the initial features reduction and the test done using the final features (in this case, using the model of London over one London test set).

Approach	True Positive Crashes	Recall	False Positive Crashes
Initial test	26 over 62 real crashes	41%	140
Final features	49 over 62 real crashes	79%	104

Fig. 7. First vs Second Features Reduction Result

Another algorithm that we tried was the K-NN classifier, but it gave bad results in any case: changing the parameter k (the number of nearest neighbours) and both using the initial datasets and the final ones (which improved the decision tree). Also, the PCA didn't improve this classifier, as it is explained in the next section.

3.5 Principal Component Analysis (PCA)

Usage of PCA, which is an unsupervised method for dimensionality reduction in data, was one of the ways to improve classification step. We tuned the parameter

responsible for the amount of obtained components, which is a transferred set of linearly uncorrelated values.

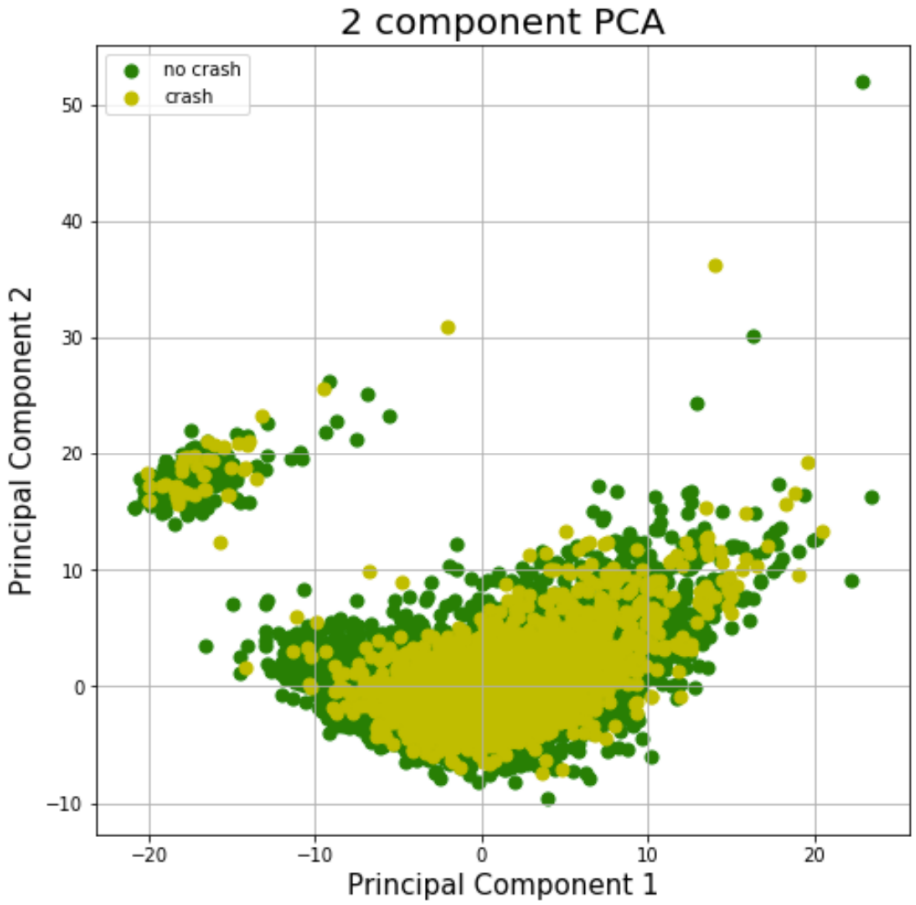

Fig. 8. Principal Component Analysis

Applying transformed data on Decision Tree, prediction model didn't provide us high results, but improved only a couple of cases that are demonstrated below, while for KNN, it never improved its classifications.

3.6 Decision Tree and Random Forest

The main approaches that we used have been the Decision Tree classifier and the Random Forest. After different attempts, the best results have been obtained using very deep trees, leaving the parameter MaxDepth to *None*. We tried to

Test (training set)	True Positive Crashes	Recall	False Positive Crashes
London (Tuscany)	36 over 62 real crashes	58%	1547
London (Rome)	22 over 62 real crashes	~35%	1059

Fig. 9. DT and kNN result

improve the models running a Random Forest algorithm, but the results have been always approximately the same. Each best decision tree has also been boosted with the Adaboost algorithm, and differently from the Random Forest, we noticed that there is on average a little improvement in the results. To validate the models, we applied the classification to the test sets. In the following table, it is possible to see an example of the results of three models predicting one test dataset of their own area (Rome model predicting a Rome test set, and so on), boosted with Adaboost.

Test/training set	True Positive Crashes	Recall	False Positive Crashes
Rome	50 over 69 real crashes	70%	72
Tuscany	74 over 112 real crashes	67%	94
London	49 over 62 real crashes	79%	104

Fig. 10. Final Result Result

In general, the results were good because each model has been able to predict, on average, 75% of the crashes happened in its test set. Considering the low number of False Positive crashes, the accuracy of models was also, on average, very high. We also used the dataset of all three areas joined together to see if we could obtain a unique model able to predict crashes in all the test sets. The results given by the decision tree which were trained with this dataset have been surprisingly good in all the test sets.

Test set	True Positive Crashes	Recall	False Positive Crashes
Rome	63 over 69 real crashes	92%	457
Tuscany	103 over 112 real crashes	91%	761
London	59 over 62 real crashes	95%	326

Fig. 11. Boossted Result

Comparing the results with the previous classifications done in separate areas, we noticed that the recall improved a lot, almost all the crashes have been predicted, but also the number of False Positive is larger. This has reduced the accuracy of predictions because of less correctly predicted non-crashes. But as we have already explained, the main purpose is to avoid crashes, giving priority to the recall. However, the reduction in accuracy is very small and this measure remained high enough (never less than 85 %, usually more than 90 %).

3.7 Multilayer Perceptron Classifier

Multilayer Perceptron is a Neural Network classifier, based on several voting layers. We trained the algorithm on the balanced datasets. Then, we trained the algorithm on the first dataset as well and compared results on test data. In all the cases, trying different training sets and parameters of the algorithm, the results have always been low for the training set, and extremely bad in the test sets. The Perceptron always classified the test set as 99/100 of *no crash*, and the few *crash*es predicted were wrong. We could conclude that in our case, this classifier came out to be useless as its results were close to a random classifier predicting only *no crash*; so, even worse than a Dummy Classifier.

3.8 Evaluating the models and Transfer Learning

Firstly, we compared our results with a dummy classifier. We tried to reproduce the real percentage of having a car crash looking at the distribution of *yes* and *no* values in *crash* columns of datasets to train a dummy classifier. All the predictions done by this model had a pretty high accuracy because of the bigger amount of *no*. However the model never correctly predicted more than 10 true positive crashes, resulting in a very low recall compared to the models we produced. In the following plot, it is possible to observe two different Precision/Recall curves, one for our classifier and another for the dummy one.

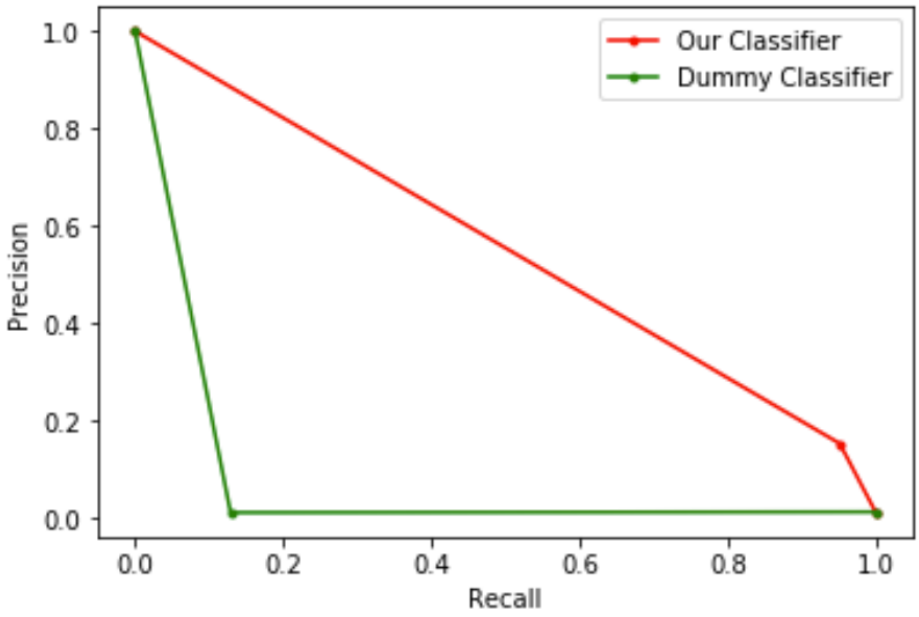

Fig. 12. Precision/Recall Curves

We tried to transfer three models to all three areas validating each model with the test set of areas different from the one where the model was trained. The purpose was to check if a model could be good enough also to predict crashes in another place. As it is possible to observe in the following tables of results, Tuscany and Rome datasets have been able to predict crashes in test sets of different areas, even if the accuracy and the recall values are lower than predicted values in their own area, while London dataset obtained totally negative results (0 true positive crashes predicted). Transferring Rome model:

Test set	True Positive Crashes	Recall	False Positive Crashes
Tuscany	38 over 112 real crashes	30%	1027
London	17 over 62 real crashes	27%	1044

Fig. 13. Model Trained On Rome

Test set	True Positive Crashes	Recall	False Positive Crashes
Rome	24 over 69 real crashes	34%	638
London	26 over 62 real crashes	41%	1082

Fig. 14. Model Trained On Tuscany

Test set	True Positive Crashes	Recall	False Positive Crashes
Rome	0 over 69 real crashes	0%	6
Tuscany	0 over 112 real crashes	0%	3

Fig. 15. Model Trained On London

Finally, we tried with the combined couple of datasets. In the following table, it is possible to see all results. Overall, three models obtained acceptable classifications, except for London testing. Still, our best model remained the one created joining all three models.

Test set	True Positive Crashes	Recall	False Positive Crashes
Rome&Tuscany on London set	17 over 62	27%	1219
Tuscany&London on Rome set	38 over 69	50%	575
Rome&London on Tuscany set	39 over 112	34%	897

Fig. 16. Model Trained On London

In conclusion, we could say that a precise classifier was obtained by joining three balanced datasets and training a decision tree, which did not result in an overfitting prediction despite of its depth.

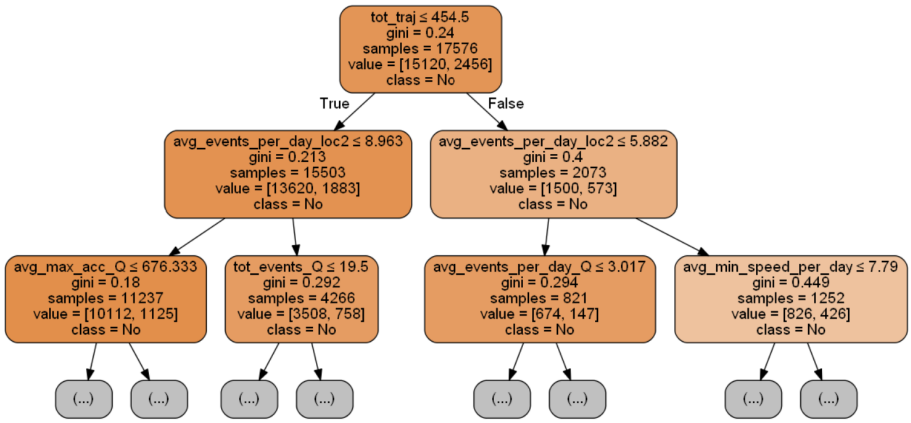

Fig. 17. Final Deciosion Tree

Despite the fact that we focused mostly on improving the recall, we computed the accuracy of the model over the training set and obtained a result of 80 % with an 8-fold cross validation.

3.9 Comparison with similar researches

We compared the model developed by our team to a model built for a similar task, namely the one created by Daniel Wilson, a Senior Data Scientist at Esri leading the GeoAI machine learning team[1]. Wilson built a model based on gradient boosting approach, which works by combining the results from many different decision trees together. It was mentioned that its model's recall rate is 89%, which is lower than the recall rate achieved in our project (92%).

4 Model Interpretation

The final decision tree that we considered as our best result is a very deep model trained on three balanced datasets joined together. It uses a large number of features to classify records. The following are the most important ones:

- tot traj: the total trajectory of the individual mobility network. avg events per day.
- loc2: the average of the number of events happened near the location 2 of the mobility network.
- evening avg speed: the average speed among all the evening travels avg.
- max acc B: the average of the max acceleration during car breaking.
- events start: the total number of times a car started a travel.
- avg avg acc Q: the average acceleration during quick cornering.
- tot events Q: the total number of times a car did a quick cornering.

– tot events B: the total number of times a car did a breaking.

Due to the number of features used by the model (114), at the beginning it appeared difficult to identify the average profile of a crashed vehicle, but we noticed that if a record had bigger values with the respect to the condition of the tree nodes, it was classified as crashed. Analyzing the most important features, we imagined that the more a car travelled (as we could see in the total trajectory or number of events happened per day features), the faster it did it (as we could see analyzing the different average speeds or accelerations attributes) and the more it was travelling in a unsafe way (for example with an high number of quick cornerings), the more it was probable for it to be classified as crashed. For example, taking into consideration the most important feature for the tree, the total trajectory, we noticed that the mean of its values among the crashed records is relevantly higher than the mean among the not-crashed records (351 for the crashed and 302 for the not-crashed ones). The same happened with the other main features, like the number of quick cornerings (the mean of crashed was 32, while for not crashed 18).

To understand better how the model was able to predict crashes, we tried to reduce it to a smaller version, using only the most important features and putting a low limit to the depth of the tree (maximum 4 levels). Furthermore, we executed a hierarchical clustering to add a column concerning the assigned cluster to each record and we discretized the values of each feature in 6 intervals to have a more uniform situation (after the discretization each record presented only values between 1 and 6 in all the columns). The simplified tree obtained is shown in the figure 18.

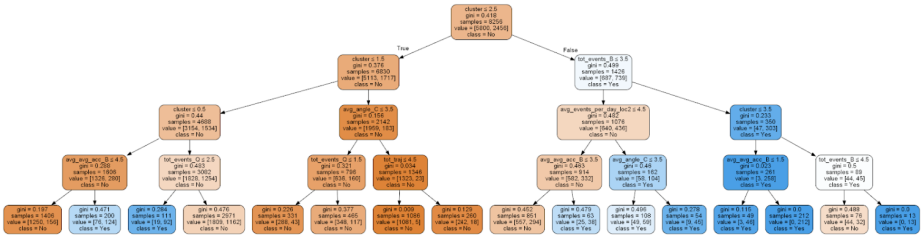

Fig. 18. Semplified Decision Tree

In this way, we were able to understand more clearly the features used to classify a record as crashed. Firstly, we noticed that the cluster column gained a big importance compared to the original complete tree. In particular, clustering managed to distinguish groups of records with a high number of crashes and groups with few crashes. To analyze each cluster, we looked at the average value of each column, and we noticed that clusters containing a high percentage of

crashes had a higher average in the feature values. For example, in the Cluster 4, where more than 95% of the records were crashes, the average value of avg events per day loc2, tot events Q, tot events B, and tot traj was higher than 5, while the average of values of the same columns in the Cluster 2, which contained a very small number of crashes, were all between 2.5 and 3 (reminding that the original values have been substituted with intervals from 1 to 6 with a discretization).

This made us conclude that a high number of unsafe driving events, like breakings or quick cornerings, and in general, a lot of kilometers travelled, are the most used features to predict a crash by the model.

We decided to provide a short classification instance to show how the decision tree operates.

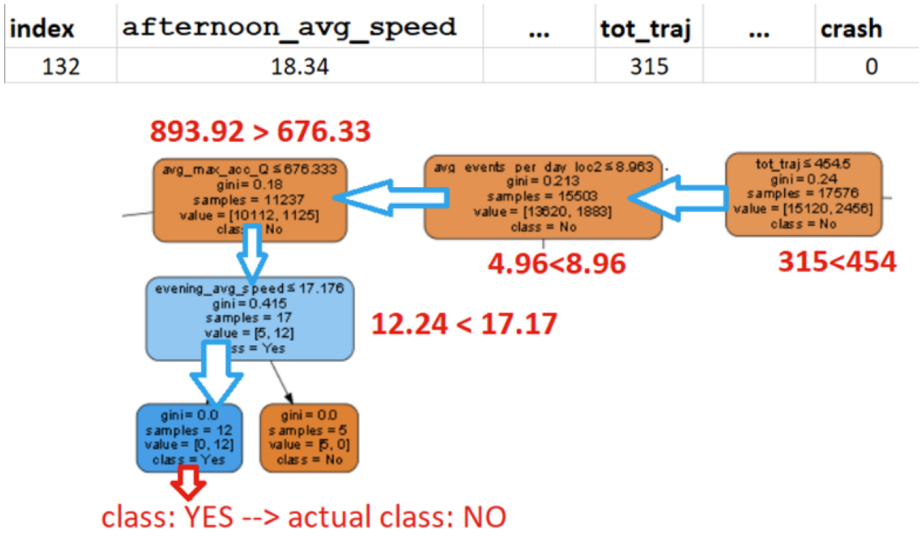

Fig. 19. Final Deciosion Tree

The example in the previous image is a wrong classification of a non-crashed record (a case of False Positive). Usually, classifications performed by this model were much deeper (in some cases, 40-45 nodes before reaching a leaf). The record taken into consideration is from a London Test set, its index is 132. As it is possible to observe, the record has values lower than the threshold for the tot traj, the first node of the tree, and the same for avg events per day loc2, avg max acc Q and evening avg speed; due to this last node, the record was classified as Yes (crashed), while actually it was No (not crashed).

Finally, considering that our model was trained on records coming from all the locations, we tried to understand if also the area could have been relevant for the tree. To do it, we added a column 'area' to each record, which registered the region to whom the record belonged. We noticed that this added feature became the most important for the model, and splitted the tree in two parts from the beginning: the very first node generated a sub-tree for London and one for the two italian areas. The problem of this re-trained model was that London had on average half of the percentage of crashed records comparing to Rome or Tuscany; for this reason, a record coming from London was easily classified as not crashed, so on the test sets this tree provided bad recall results.

5 Conclusion

To reach the results obtained, we used 7 training sets provided for each area, London, Tuscany and Rome. After the data understanding phase, we tried different approaches oversampling the data and to reducing the amount of features. The following best approach was created: for each area, we took all the crash records and randomly add no crash records from all the seven datasets.

To classify a car crash, supervised methods were observed. To train different models, we implemented several classification algorithms designed by Python. We used each individual area dataset and all their possible combinations to test 21 different test sets. The application of Principal Component Analysis, converting large amount of features into a set of values of linearly uncorrelated variables, improved just few cases. That is why we could conclude that the usage of PCA wasn't significant. After trying to remove columns through correlation, the best results were reached by considering 114 features of our choice.

As it is possible to see in table 10, most of the records were predicted correctly by the combined training data set. The obtained recall was never less than 91% , with an accuracy never less than 85%. We assumed that the success was mainly the result of the amount and variety of training instances. Surprisingly, a decision tree classifier provided us the highest results. The labels of the tree allowed us to analyze the model explanation step easier.

Following the idea of importance of car crash prediction, we focused on feature importance. We were interested in providing both safe and unsafe driving styles. Additionally, a clustering step was implemented to add a column that could have been useful for classification. However, the results didn't lead us to interesting observations or conclusions. We noticed a slight tendency of having greater values than averages for unsafe drivers. This means that the individual mobility network of crashed vehicles, on average, had a bigger total trajectory, higher speed, higher acceleration and higher number of unsafe behaviours (like quick cornering) than the mean of these values for not crashed mobility networks. In conclusion, finding a model able to predict car crashes is an important task both for economic and

ethical reasons. The models obtained gave us very high results in the case we took into consideration (21 datasets about London, Tuscany and Rome). The hardest task would be obtaining a good model trained in a certain situation and applied to many different ones. In our case, the only situation that reached good results in this transfer learning goal was the model trained over the combination of all the cities, so it would be interesting to test it over new areas and sets of individual mobility networks.

The limitation of our best model is complexity as the final decision tree was very deep (In some sub-trees reaching 40-45 levels of depth), which makes a possible analysis concerning crashed records identity difficult. Moreover, the model needs a specific and big group of features to create an individual mobility network for each vehicle.

References

1. Wilson, D. (2018). Using Machine Learning to Predict Car Accident Risk. Retrieved from: https://medium.com/geoai/using-machine-learning-to-predict-car-accident-risk-4d92c91a7d57
2. Chen W., Lin L., Chengcheng X., Weitao L., (2019). Predicting Future Driving Risk of Crash-Involved Drivers Based on a Systematic Machine Learning Framework. Retireved from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6388263/
3. Rinzivillo S., Gabrielli L., Nanni M., Pappalardo L., Pedreschi D., Giannotti F. (2014). The purpose of motion: Learning activities from Individual Mobility Networks. Retrieved from: https://ieeexplore.ieee.org/document/7058090