UNIVERSITÀ DI PISA

Data mining and Machine Learning

ACADEMIC YEAR 2022-23

Maria Fabijan

1. Introduction

The Leeds is the city in United Kingdom with almost one million inhabitants. In reascent years there was a lot of car accident caused by different, but correlated reasons. These intercessions such as weather, location, road type and conditions are searched to develop classifier, which can bring the information if the driver are safety or if he/she should choose another type of communication.

2. The goal

The main goal of the project was to create a classifier for preventing car accidents in Leeds. Entering data such as:

- Number of Vehicles
- Time (24hr)
- Road Surface
- Lighting Conditions
- Weather Conditions
- Type of Vehicle
- Casualty Class
- Casualty Severity
- Sex of Casualty
- Age of Casualty
- Grid Ref: Easting and North

The driver can check if his/her way will be safety or if he/she should choose another type of communication.

3. The dataset

The dataset comes from ***The Publications Office of the European Union***, which is the official provider of publishing services to all EU institutions, bodies, and agencies. As such, it is a central point of access to EU law, publications, open data, research results, procurement notices and other official information.

The data used for this project is from 2014, 2015, 2016, 2017, 2018 and 2019, which come in various forms. This dataset was gathered in one dataset.
https://data.europa.eu/data/datasets/road-traffic-accidents?locale=pl

Entire dataset used for classification contains 24 columns and 11187 rows.

4. Preprocessing

3.1. Load and merge datasets

The dataset was collected from several years, thus the first step was combining the datasets into one set.

3.2. Clean dataset

In several columns the values represented the same variable have the different name data type, therefore the next set was changing string data type for float type, using

accident guidance. Moreover, I had to check the remaining string values, which were not defined in guidance and then I repeated exchanges. The missing values was denoted as some values, wherefore I verified these null values and denoted them as None.
There are several columns, which are not providing relevant information or consist only few samples, then I delete them.

3.3. Data plot

I used histogram to plot the distribution to get a preliminary idea of the distribution the data follows. Moreover, I exploit the boxplot to get insight for the outliers. The last data display refers to class distribution. The dataset is imbalanced. It consists less than 1% of (1) Fatal class, 14% of (2) Serious class and almost 85% of (3) Slight Casualty Severity.

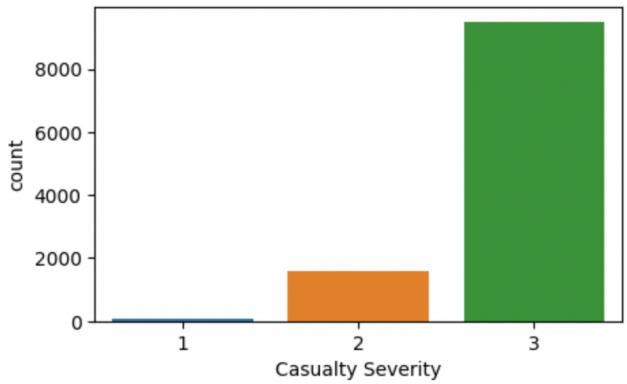

Figure 1: Class distribution

3.4. Clustering

I used K-means algorithm with 1000 clusters to merge the grid-ref parameters and obtain more consistent locations.

5. Classification

I used several methods imported from scikit-learn library to create the classifiers. Each method was used three times on various data, such as originally data, oversampling data and undersampling data. The highest dangerous is represented by Fatal Casualty Severity (1 class), thus I mainly focused on metrics measured on this case, especially on F1 score, which is the harmonic mean of precision and recall and takes into account both false positive and false negatives.

For each model I used 10-fold cross-validation procedure, which is a technique used for model evaluation and selection. In this method, the data is divided into "k" subsets (folds), where "k - 1" subsets are used for training the model, and the remaining subset is used for testing. This process is repeated "k" times, each time with a different subset for testing and the average performance of the model is calculated from the performance of each iteration. This provides a better estimate of the model's performance compared to a single train-test split as all the data is used for both training and testing. Furthermore, I exploited 5-fold cross-validation to tune the hyperparameters of a model, as it provides an estimate of the performance of the model for different combinations of hyperparameters.

4.1. Decision tree

The tree is constructed by recursively splitting the data into subsets based on the feature that provides the highest information gain, i.e., it best separates the target variable. The process is repeated until a stopping criterion is met, such as a minimum number of samples required to split a node, or a maximum depth of the tree. The final prediction is made by taking a majority vote of the training examples in the leaf node the test data falls into.

The first models, that I created was based on Decision Tree method. The best results were achieved by Decision Tree based on original data. For improve the model I used grid search to find the best hyperparameters. Eventually the best model was the Decision Tree trained on original data with criterion as entropy, min_samples_leaf = 2 and min_samples_split=2.

```
accuracy 0.7495633136174581
```

	1	2	3	macro avg	weighted avg
f1-score	0.137387	0.257741	0.853328	0.416152	0.761775
precision	0.105981	0.233869	0.874603	0.404818	0.776126
recall	0.201111	0.288015	0.833170	0.440765	0.749563
support	9.500000	155.900000	923.100000	1088.500000	1088.500000

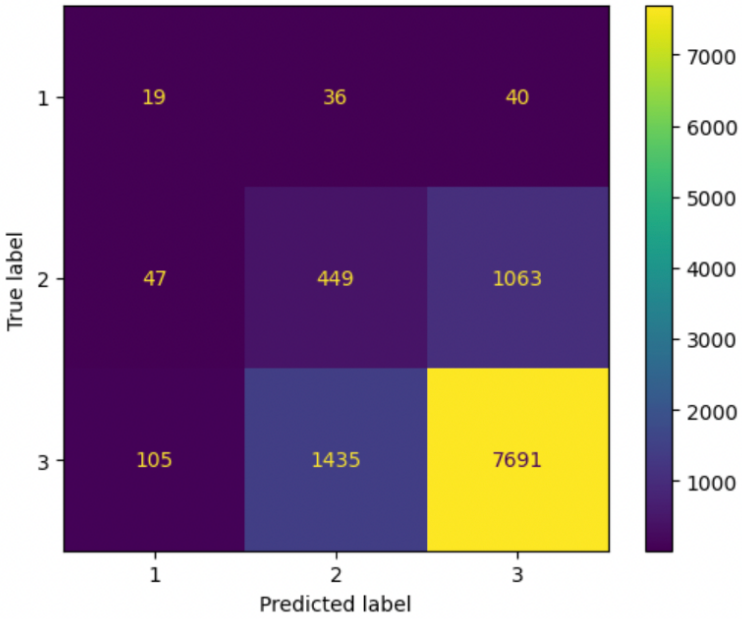

Figure 2: The metrics of best Decision Tree model.

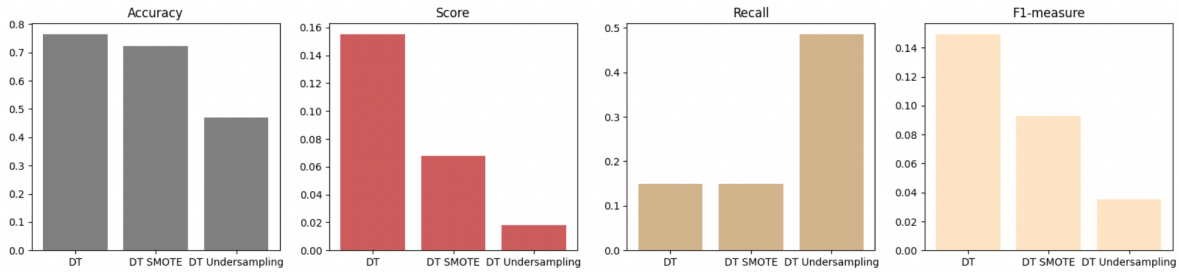

Figure 3: Comparison of Decision Tree models metrics

4.2. Random Forest

It's a combination of multiple decision trees, where each tree is trained on a random subset of the data, and the final prediction is made by aggregating the predictions of all the trees.

In Random Forest, each decision tree is trained independently and makes predictions based on the features and conditions in the data. At each split in the tree, only a random subset of features is considered for splitting, instead of considering all features as in a normal decision tree. This process is repeated multiple times to create multiple decision trees, and the final prediction is made by taking the average or majority vote of all the trees.

To extend Decision tree algorithm I used Random Forest method. Similar to Decision Tree model, the best model of Random Forest was obtained by training on unsampled data.

accuracy 0.7177761910549344

	1	2	3	macro avg	weighted avg
f1-score	0.092493	0.280602	0.829852	0.400982	0.744754
precision	0.064613	0.231703	0.880812	0.392376	0.780725
recall	0.167778	0.356046	0.784530	0.436118	0.717776
support	9.500000	155.900000	923.100000	1088.500000	1088.500000

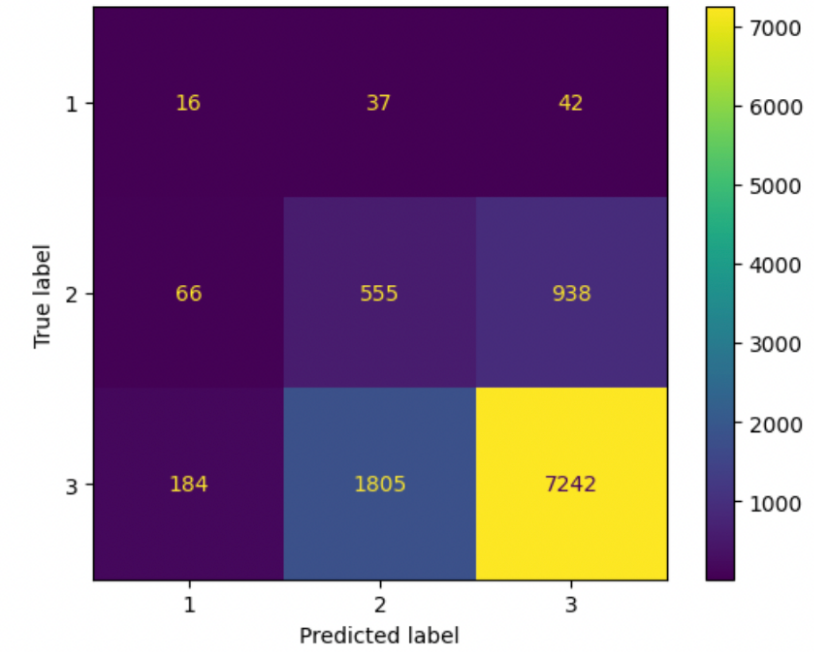

Figure 4: The metrics of best Random Tree model

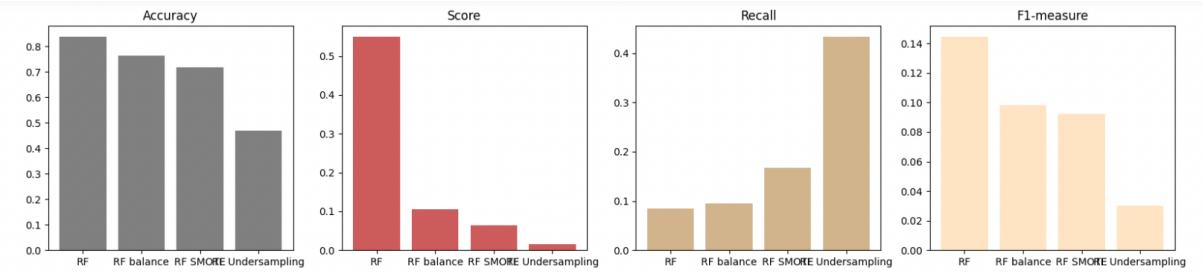

Figure 5: Comparison of Random Tree models metrics

4.3. Naïve Bayes

Naive Bayes is a simple probabilistic algorithm that makes class predictions based on Bayes' theorem. There are two main types of Naive Bayes algorithms: Gaussian Naive Bayes and Multinomial Naive Bayes. I used the Multinomial Naive Bayes, because I the data is not normally distributed.

	1	2	3	macro avg	weighted avg
f1-score	0.020235	0.208819	0.545074	0.258043	0.492340
precision	0.010599	0.147162	0.862756	0.340172	0.752826
recall	0.243333	0.396402	0.404185	0.347973	0.401659
support	9.500000	155.900000	923.100000	1088.500000	1088.500000

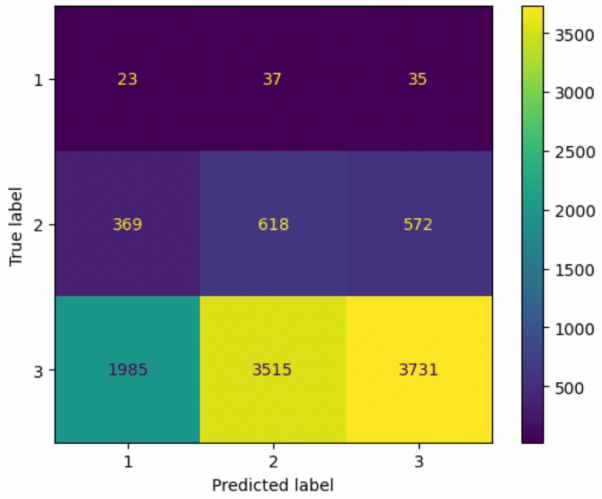

Figure 6: The metrics of best Naïve Bayes

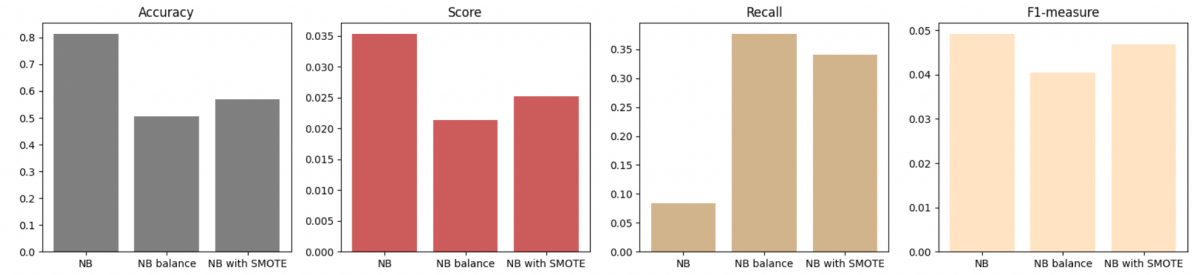

Figure 7: Comparison of Naïve Bayes models metrics

4.4. Logistic Regression

Softmax Regression is a generalization of Logistic Regression to handle multi-class problems. In this technique, the predicted probabilities for each class are calculated using the softmax function, which maps the input values to a probability distribution over all classes. The class with the highest predicted probability is then selected as the final prediction.

accuracy 0.5098770120996057

	1	2	3	macro avg	weighted avg
f1-score	0.031842	0.242503	0.674097	0.316147	0.606677
precision	0.016535	0.186734	0.902181	0.368483	0.791984
recall	0.431111	0.346394	0.538297	0.438601	0.509877
support	9.500000	155.900000	923.100000	1088.500000	1088.500000

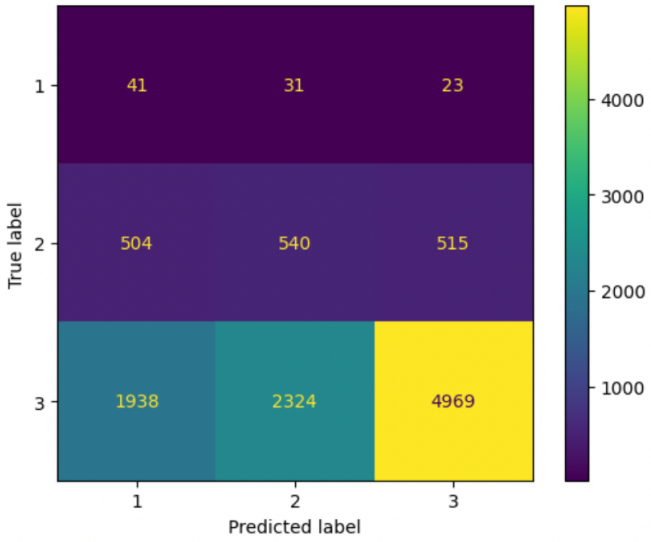

Figure 8: The metrics of best Logical Regression model

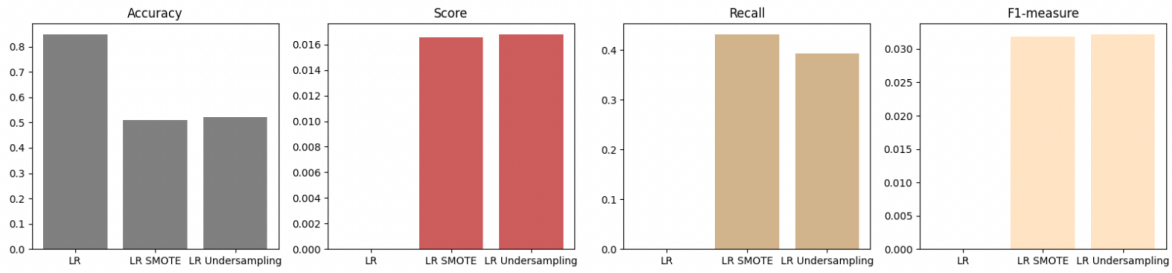

Figure 9: Comparison of Logistic Regression models metrics

6. Finally model

Based on the metrics provided, it appears that the Random Forest model performed well for the classification task with an unbalanced class. The weighted average of precision, recall, and f1-score are all high, indicating that the model was able to accurately predict the target class for the majority of the data. In particular, the recall score of 0.837759 suggests that the model was able to detect the target class with a high degree of accuracy.

It's worth noting that while the weighted average metrics appear to be strong, the performance of the model on individual classes might be more uneven. The low f1-score and precision for classes 1 and 2 indicate that the model may not be performing

as well on these classes, due to the imbalance in the data. However, the score on the minority class was achieved by this model.

In conclusion, the Random Forest model appears to be a good choice for this classification task with an unbalanced class, based on its overall performance as indicated by the weighted average metrics. However, it's important to consider the performance of the model on individual classes, especially those with imbalanced data, to ensure that the results are meaningful and reliable.

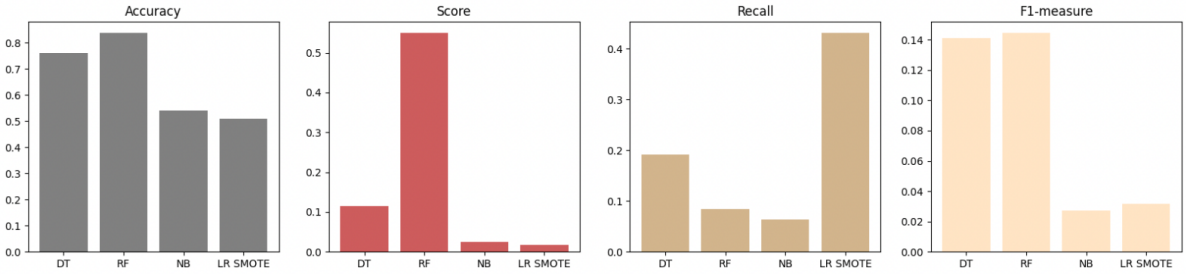

Figure 10: Comparison of all the best models