ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования

Национальный исследовательский университет
«Высшая школа экономики»

Факультет экономических наук
Образовательная программа «Экономика»

КУРСОВАЯ РАБОТА

«Дихотомический выбор, Агрегирование голосов и теорема Кондорсе»
«Dichotomous Choice, Aggregation of Votes and Condorcet's Jury Theorem»

Студент группы БЭК181
Щекотов Иван Сергеевич, Тамогашев Кирилл

Научный руководитель:
Борис Демешев

Москва 2021

Contents

1 Introduction

2 Literature Review

3 Data Preprocessing and Description

For our purpose we have chosen data gathered from January 2015 to April 2021 in Moscow. Data is represented via `geojson` format and contains extensive amount of features describing various aspects of car accidents, including coordinates, lighting, weather and road conditions, nearby objects, datetime, severity, information about vehicles, drivers and passengers involved, their health conditions in the aftermath, rule violations that caused accident, injured and dead counts[1]. Almost all of the described features fall into the categorical type and are of nested structure (i.e. there are multiple cars, people, road conditions in an accident) and they are non-hierarchical, which means simple label encoding is not applicable.

3.1 Generalizing and encoding unique values

The problem with data is that due to it's nature, some of the features have an enormous amount of unique values which are not relevant in all their variety. Our methodology insists on merging these unique values for every such category into groups by some inherent property that these values possess, e.g. if a person dies after being hospitalized, we prefer not to distinguish between various time spans over which death has occured or if there are various types of passenger cars, there is no reason to analyze them as being different. In another words we try to explicitly project data into lower dimension, because it is impossible to work with otherwise. We discuss these features further and describe ideas which led us to unification. All code that refers to generalizing unique values is presented in `utils` folder[2].

3.1.1 Violations[3]

There are 104 unique violations that were ascribed to accidents. Violations are commited by drivers or by pedestrians (people that are not driving behind the wheel of the vehicle and that are not passengers) and they can be divided into 8 groups:

1. violations of driving with respect to car motion
2. violations of goods transportation or carriage of passengers
3. violations of obligations or non-compliance when driving a motorcycle
4. improper use of light signals to control traffic
5. violations commited by pedestrians
6. non-compliance with rules of safety when driving
7. violations of vehicle operation
8. other

There is no unified classification provided by Russian department of transport, so our division may be incorrect. It is based on sane reasoning and similarities between different events.

[1] Example of how data is structured can be found here.
[2] https://github.com/isdevnull/cw3/tree/dev/utils
[3] https://github.com/isdevnull/cw3/blob/dev/utils/violations.py

3.1.2 Nearby Objects[4]

Nearby objects represent buildings surrounding an accident and road type (i.e. junctions, pedestrian crossings, etc.). There are 58 unique values and they can be divided into 8 groups:

1. other
2. unmarked and marked junctions
3. unmarked and marked pedestrian crossings
4. places with increased transport density (usually, some kind of stop on road)
5. crowded places (e.g. bus stop)
6. controlled junctions
7. controlled pedestrian crossings

'Other' mostly contains various types of buildings, while other groups relate to some road objects. We didn't break 'other' into multiple groups because considered this step irrelevant for our purposes or reckoned that groups are rather small to be represented (both by accident occurence and unique values).

3.1.3 Automobile Categories[5]

Different types of vehicles (82) were divided into the following groups:

1. passenger cars
2. elite passenger cars
3. trucks
4. public transport
5. offroad and heavy-duty vehicles
6. motor vehicles with less than 4 wheels
7. other

3.1.4 Health Status[6]

The consequences of accidents are characterized by different health status (41). We define the following groups both for drivers, passengers and pedestrians:

1. no injuries
2. light injuries
3. wounded victims
4. death occured before being transported to hospital
5. death occured after receiving treatment in hospital

3.1.5 Preprocessing[7]

Let $F = \{\text{'violations'}, \text{'status'}, \text{'transport'}, \text{'nearby'}\}$ – features to be transformed.

Let S be a set of all available features. $C \subset S \setminus F$ – categorical features to be encoded.

Let E be the space of all possible feature encodings.

Let $g \colon F \hookrightarrow E$ – injective function that applies feature specific encoding.

Let $h \colon C \hookrightarrow \mathbb{N}$ – function that returns number of unique elements for given feature.

Let D be our sample of accidents.

[4] https://github.com/isdevnull/cw3/blob/dev/utils/nearby.py
[5] https://github.com/isdevnull/cw3/blob/dev/utils/transport.py
[6] https://github.com/isdevnull/cw3/blob/dev/utils/health_status.py
[7] https://github.com/isdevnull/cw3/blob/dev/preprocessing.py

When we apply some function to the feature, we assume that feature is a column of size of all our data, e.g. 'violations' $\in X^{|D|}$, where X is a set of unique violations. This is done to remove multiple for-loop nesting over all data points when describing algorithm.

Preprocessing algorithm

Initialize: $F_g \leftarrow \{\}$
for $x \in F$ **do**
    $F_g \leftarrow F_g \cup g(x)$
end for
$C \leftarrow C \cup F_g$
for $c \in C$ **do**
    $T_c \leftarrow one\_hot\_encoding(c)$, where $T_c = \left\{t_1, \ldots, t_{h(c)}\right\},\ t_j \in \{0,1\}^{|D|},\ j \in \{1, \ldots, h(c)\}$
    $C \leftarrow C \setminus c$
    $C \leftarrow C \cup T_c$
end for

This is a general procedure which happens when we preprocess our data in `download_and_preprocess_data()`. Eventually, we get a dataset with all the features that we want to explore. Some features like 'driver experience' are extracted from nested structure and put in a list because there can be multiple drivers involved in an accident. We also extract percent of women involved in an accident as driver, e.g. if there was a collision between a man driver and a woman driver, then the value would be 0.5. Lastly, we select only those accidents that occured in Moscow. Initial data has some flawed points that need to be removed. That is why we use the following bounding conditions on latitude: from 55.1339600° to 55.9825000°, longitude: 37.1813900° to 37.9545100°, which correspond to Mihnevo (55.1339600°, 37.9545100°) and Zelenograd (55.9825000°, 37.1813900°). Data is saved in `csv` format and is accessible from root of repository. The code is applicable (maybe with minor tweaks) to all other Russian regions.

References

[1] Проект «Карта ДТП»