Car Accident Severity Capstone Project (Week 2)

Table of contents

Introduction: Description of the Problem

In this project a model to predict the **probability** of having a **severe car accident** taking into account the conditions of the road will be build up. In order to do so, the city of **Seattle** will be used as an **example**. A database containing information about car accidents in Seattle will be used to feed the model.

The **stakeholders** of this project will be **drivers** using their vehicles in Seattle and its surroundings. Of course, this project may also interest app developers which may use this example and extrapolate it to other cities and/or states.

The main idea behind this project is quite simple: "***forcasting car accidents is the best way to avoid them***".

Data

The data that will be used in this project comes from the ***Traffic Management Division of Seattle***. A raw database in a form of a CSV file will be downloaded from: "https://s3.us.cloud-object-storage.appdomain.cloud/cf-courses-data/CognitiveClass/DP0701EN/version-2/Data-Collisions.csv". This **database contains** precious information about all registered **car accidents in Seattle from 2004 to the present**.

Understanding the Data

The raw database comes as a **CSV file** containing **194673** entries (car accidents in Seattle) and **38** attributes with several redundant data (columns) that will not be used to build the model. The target of our model will be the "**severity**" of the possible car accident.

The problem has been already simplified; the raw database has only **two different values to describe the severity** (see **Figure 1**):

1) **Property Damage Only Collision** 136485 accidents representing *ca*. 70% of the data.

2) **Injury Collision** 58188 accidents representing *ca*. 30% of the data.

SEVERITYDESC	
Property Damage Only Collision	136485
Injury Collision	58188

SEVERITYCODE	
1	136485
2	58188

Figure 1: Type of car accident severity describe in the Seattle car accident database.

The dataset includes the following fields:

SEVERITYCODE	int64
X	float64
Y	float64
OBJECTID	int64
INCKEY	int64
COLDETKEY	int64
REPORTNO	object
STATUS	object
ADDRTYPE	object
INTKEY	float64
LOCATION	object
EXCEPTRSNCODE	object
EXCEPTRSNDESC	object
SEVERITYCODE.1	int64
SEVERITYDESC	object
COLLISIONTYPE	object

PERSONCOUNT	int64
PEDCOUNT	int64
PEDCYLCOUNT	int64
VEHCOUNT	int64
INCDATE	object
INCDTTM	object
JUNCTIONTYPE	object
SDOT_COLCODE	int64
SDOT_COLDESC	object
INATTENTIONIND	object
UNDERINFL	object
WEATHER	object
ROADCOND	object
LIGHTCOND	object
PEDROWNOTGRNT	object
SDOTCOLNUM	float64
SPEEDING	object

ST_COLCODE	object
ST_COLDESC	object
SEGLANEKEY	int64
CROSSWALKKEY	int64
HITPARKEDCAR	object

Figure 2: Fields of the Seattle car accident database.

Only few of the database attributes (columns) are relevant for the prediction of the probability of having a severe car accident, namely:

- the condition of the road during the accident (**ROADCOND** attribute)
- the weather conditions during the time of the accident (**WEATHER** attribute)
- the light conditions during the accident (**LIGHTCOND** attribute)
- category of junction at which the accident took place (**ADDRTYPE** attribute)
- longitude (**X** attribute)
- latitude (**Y** latitude)

As a starting point, the attributes above will be used as main independent variables to build up our predicting model. Other parameters such as the **time of the accident** (**INCDTTM**) may also be included in future models in order to explore possible improvements.

During the dataset preparation and cleaning, in order to remove possible biases in our model some of the entries containing positive (true) parameters such as **inattention** (**INATTENTIONIND** attribute), **speeding** (**SPEEDING** attribute) or **drug influence** (**UNDERINFL** attribute) will be removed.

Evolution of the Car Accidents in Time

One of the first things that we need to verify is the evolution of accidents in time. In principle, over the years the quality of the streets, roads and even the cars themself may have improved, thus, driving might have become safer and less accident may have happened in the last years. We can verify this hypothesis looking at the trend of car accidents in the last few years with a horizontal bar chart.

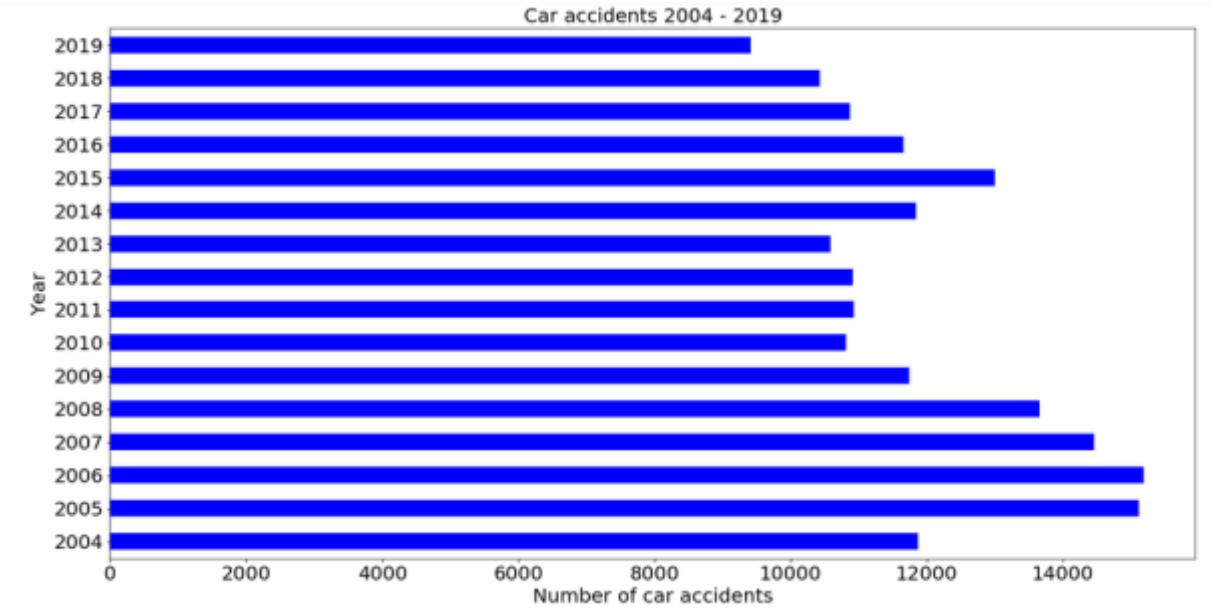

Figure 3: Evolution of car accidents in Seattle from 2004 until 2019.

As it can be observed in the horizontal bar chart above, the car accidents have decreased in the last decade. However, this decreasing trend might not be significant enough to discard the entries from the 2000's decade; thus, all the years will be taken into account in order to construct the predicting severity model.

Density of Car Accidents by Districts

Another important trend that can be explored is the density of car accidents in each district. In order to do so, the first thing that needs to be done is a transformation of latitude and longitude coordinates into zipcodes. To transform the latitude and longitude coordinates the GeoPy client (https://github.com/geopy/geopy) will be used.

At this point only the X and Y attributes will be used. However, some precautions need to be made, we need to verify if there are some instances with blank data and in this case we need to delete them in order to make the transforming function ("get_zipcode") work properly. Indeed, after verification we find that **5334** entries have no latitude and longitude coordinates. Thus we need to remove them from the dataset that will be used to construct the model.

Only a small part of the dataset (**the first three hundred entries**) **will be used** in order **to analyze the density of car accidents** in each districts (represented by the zipcodes), since the get_zipcode function needs to make calls to external APIs for each pair of coordinates. Thus, in order to transform the whole dataset into zipcodes the required time would be several days.

After examining the results of the "get_zipcodes" function, only 4 entries had to be removed since the zipcodes found: 98106-1499, 98133-6124 and 98109-5210 had more than 5 digits. The 296 remaining instances of car accidents in Seattle are represented in **Figure 4**:

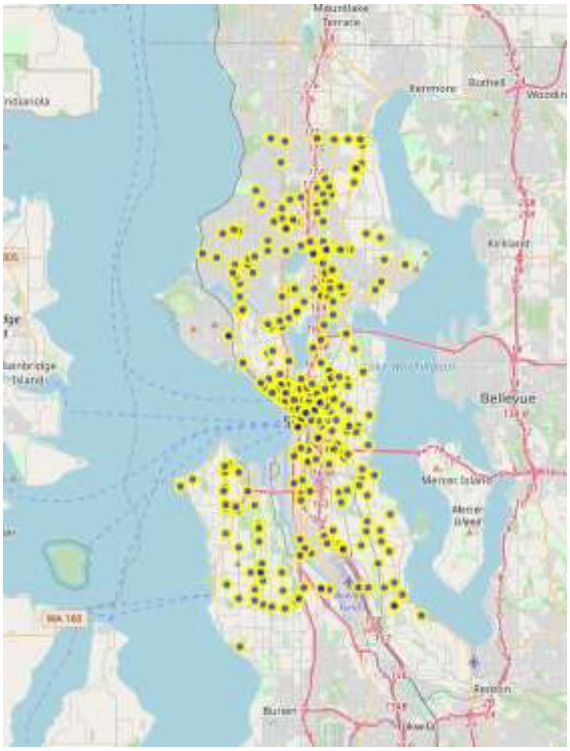

Figure 4: Sample of 296 car accidents in Seattle (from 2004 to 2020).

In order to represent the 296 car accidents in the map, the Folium Python library has been used.

When representing the density of car accidents by “districts” (**Figure 5**) it can be seen that most of the accidents have happened in the east-center part of the city and one of the north-east districts.

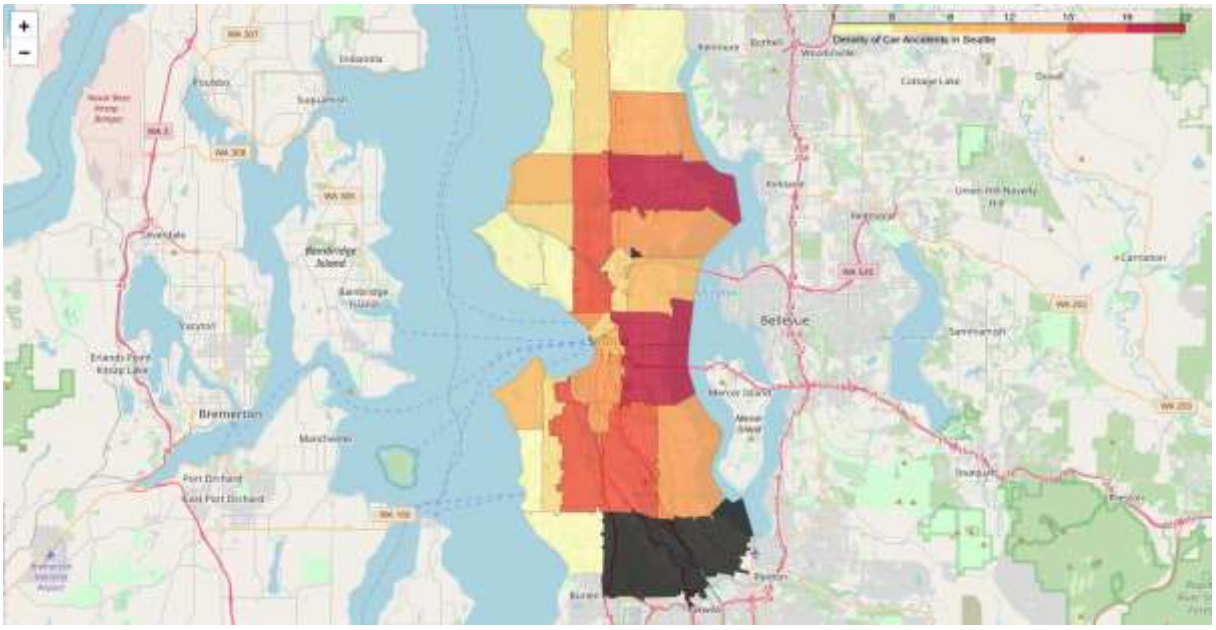

Figure 5: Density of car accidents in Seattle (from 2004 to 2020) by zipcodes.

Nevertheless, comparing both images (**Figures 4** and **5**), one can clearly see that the zipcodes might not be the best way of representing the car accident **densities** in the maps, thus, a clustering method will be investigated.

Cleaning Data

Now the data needs to be cleaned before been used to build up our model. All the entries containing “Unknown”, “Other” and/or empty will be erased from the dataset that will be used to build the model. Only the attributes that are interesting for the study (**ROADCOND**, **WEATHER**, **LIGHTCOND**, **ADDRTYPE**) will be checked and treated.

Methodology

In order to predict the severity of a possible car accident in the city of Seattle three different supervised machine learning algorithms will be used to build our model:

1) **Support Vector Machine** (**SVM**),

2) **Decision Tree** (**DT**) and

3) **Logistic Regression** (**LR**)

The advantage of the latter machine learning algorithm (***Logistic Regression***) is that it **can predict** not only a **categorical dependent variable** (the severity of a car accident in this project) but also **its probability**.

All the models will be trained and tested in order to evaluate their performance and to determine which model is the best one.

As seen in the "***Density of Car Accidents by Districts*****"** section, transforming the coordinates into zipcodes to represent the car accident densities may introduced some biased information in our dataset. In order to include meaningful information about the "position" of the car accidents, the unsupervised machine learning algorithm known as "**Density-Based Spatial Clustering of Applications with Noise (DBSCAN)** can be used. This algorithm developed by Martin Ester, Hans-Peter Kriegel, Jörg Sander and Xiaowei Xu in 1996 is one of the most common clustering algorithms which works based on the density of the object. The main idea behind this unsupervised algorithm is that if a particular point belongs to a cluster, it should be near to lots of other points in that cluster. In principle, clusters based on the "real" car accident densities instead of the zipcodes will be computed.

Concerning the location of the accidents a hypothesis is made:

"*Though including the "position" of the car accidents into the model may seem an obvious and necessary choice, the* ***ADDRTYPE*** *attribute may already contain enough information about the road and thus the "location" of the accident. Therefore, the "position" of the accidents may be redundant and useless information.*"

In order to verify this hypothesis all the models will be built up with and without the "position" information.

Analysis

In order to have a better idea of the car accident densities in a real map, let's superimpose the locations (latitude and longitude). This procedure can be done by different methods such as the use of the **Basemap** package that can be used together with the library **Matplotlib** or directly with the **Folium** library as previously done. This time the full cleaned dataset will be used (not only the first 300 instances). Once again, the Folium library will be used. Let's begin with the plot of all car accidents superimposed into the map as small red circles (see **Figure 6**).

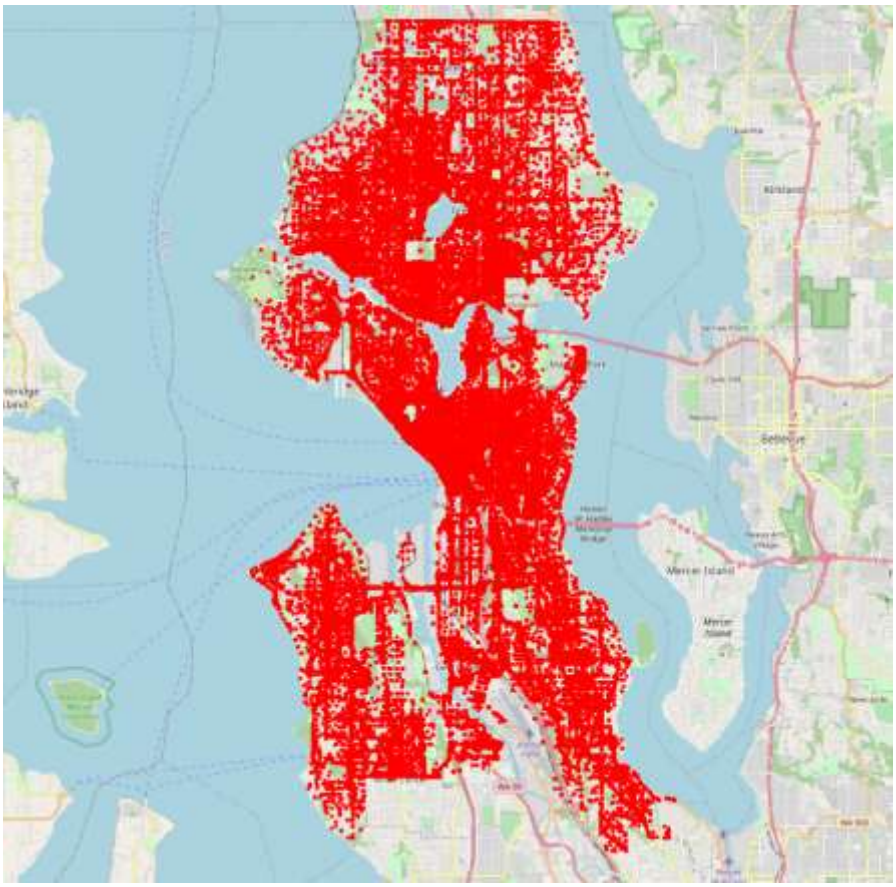

Figure 6: All car accidents in Seattle (from 2004 to 2020).

As it can be observed in **Figure 6**, a very high density can be found in the middle of the city and as we get close to the outskirts of the city the density slightly decreases. However, the car accident density is so high that almost every part of the city is colored in red. In order to have a clearer global vision of the real densities, let's use the clustering method implemented in the folium library as an initial approach. It is important to notice that this clustering method is not the one implemented in **Scikit-learn** that we will use later on to determine the different clusters via the **DBSCAN** algorithm.

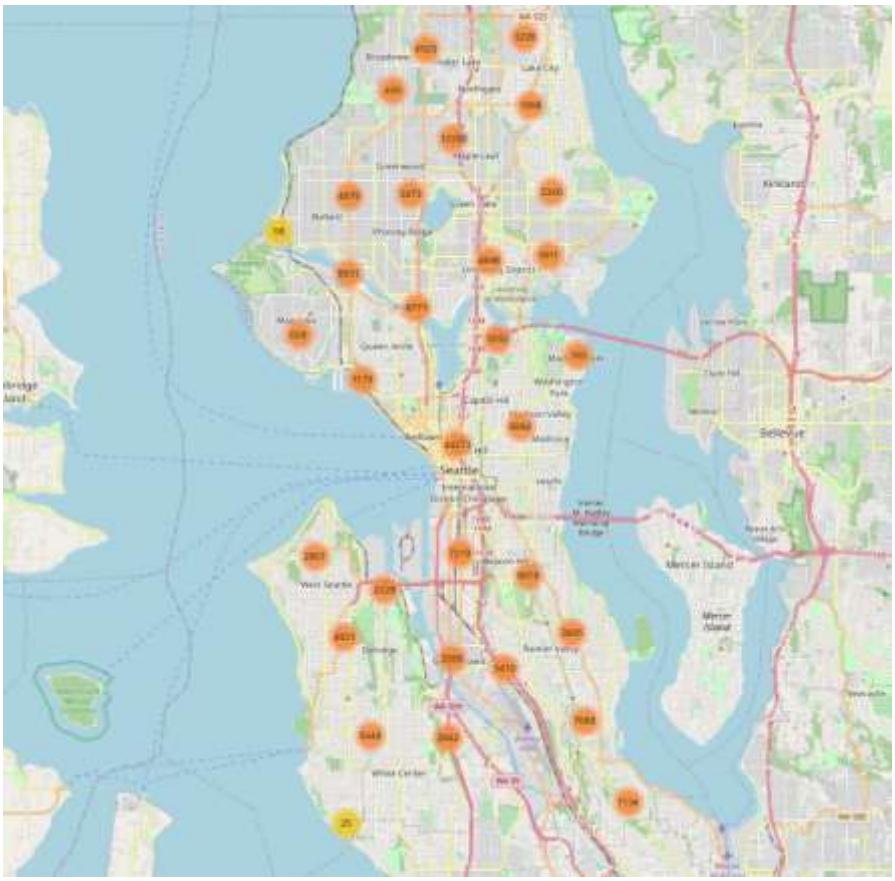

Figure 7: All car accidents in Seattle (from 2004 to 2020) grouped in clusters.

From the picture above (**Figure 7**), it can be clearly seen that the higher density of car accidents can be found in the very center of Seattle near the **Belltown district** and also in the north near the **Maple Leaf district**.

Visualization of Clusters Based on Location

Let's use now the **Scikit-learn DBSCAN** algorithm to generate the so-called density-based clusters. The DBSCAN algorithm requires only two parameters:

1) **Epsilon**: the radius defining all the neighbors of the same cluster and
2) **minimumSamples**: the minimum number of data points needed to define a cluster.

We will start with an Epsilon = 0.15 and a minimumSamples = 10.

Apparently with these initial values we get only 1 cluster and only one outlier labeled by -1. Obviously, this is not a satisfactory partition, thus, other parameters will be used. With Epsilon = 0.03 and a minimumSamples = 10 we obtain **447** clusters and **4280** outliers.

Finally, the colored clusters can be plotted into the map (see **Figure 8**).

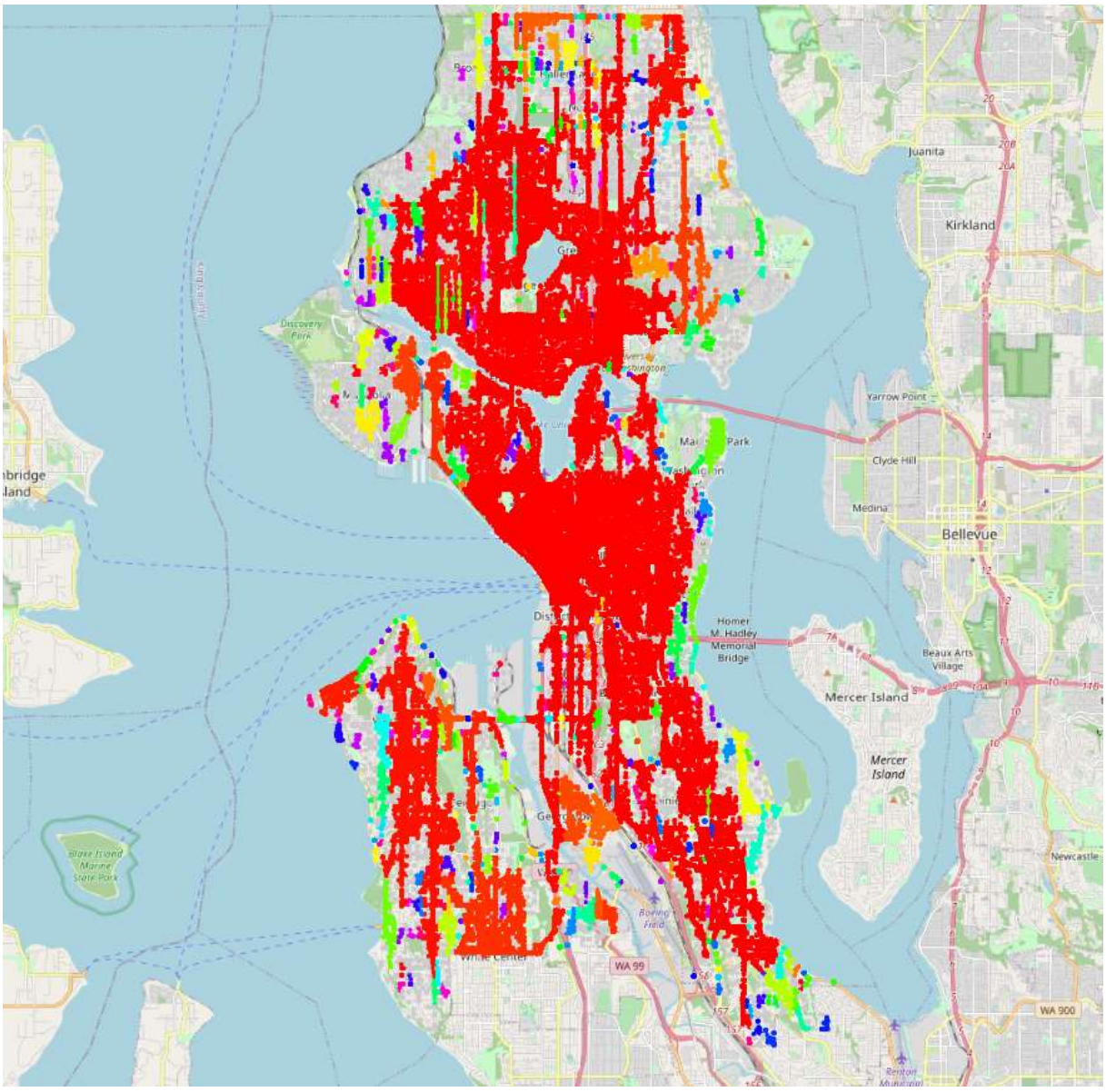

Figure 8: The 447 clusters found by the density of car accidents in Seattle (from 2004 to 2020).

We build a model with the clusters obtained by the unsupervised DBSCAN machine learning algorithm and we remove all the outliers.

Converting Categorical Features into Numerical Values

Let's analyse the **ROADCOND**, **WEATHER**, **LIGHTCOND** and **ADDRTYPE** attributes:

	ROADCOND
Dry	116019
Wet	43965
Ice	986
Snow/Slush	768
Standing Water	93
Oil	49
Sand/Mud/Dirt	49

Figure 9: Categorical Values of the "ROADCOND" feature.

	WEATHER
Clear	103885
Raining	30884
Overcast	25692
Snowing	782
Fog/Smog/Smoke	514
Sleet/Hail/Freezing Rain	103
Blowing Sand/Dirt	41
Severe Crosswind	23
Partly Cloudy	5

Figure 10: Categorical Values of the "WEATHER" feature.

	LIGHTCOND
Daylight	107739
Dark - Street Lights On	44327
Dusk	5384
Dawn	2267
Dark - No Street Lights	1186
Dark - Street Lights Off	1026

Figure 10: Categorical Values of the "LIGHTCOND" feature.

	ADDRTYPE
Block	102107
Intersection	59822

Figure 11: Categorical Values of the "ADDRTYPE" feature.

We need to transform all these categorical values into numerical values so that they can be interpreted by the different **Scikit-learn** models.

We will define a new dataframe called "**features**" containing all the attributes that will be used to build the different machine learning models.

	ROADCOND	WEATHER	LIGHTCOND	ADDRTYPE	Clus_Db
0	1	2	0	1	0.0
1	1	1	1	0	1.0
2	0	2	0	0	0.0
3	0	0	0	0	0.0
4	1	1	0	1	0.0

Figure 12: Numerical features used to build the three different machine learning models.

Results and Discussion

The first thing that needs to be done is splitting the dataset into the train and test parts. The test set contains **24290** entries and the training set contains **137639** entries. **15 %** of the full data set has been used to generate the test set. Before using the training and test sets they need to be normalized independently.

Initially, the transformed numerical features **ROADCOND**, **WEATHER**, **LIGHTCOND** and **ADDRTYPE** together with the generated **Clus_Db** will be used to predict the severity of a possible car accident in the city of Seattle. The latter feature contains information about the "position" of the car accident in a cartographic map of Seattle and has been successfully transformed into density-based cluster information by the DBSCAN algorithm as implemented in the **Scikit-learn** python library. Three different models using: 1) the **Support Vector Machine** algorithm, 2) the **Decision Tree** algorithm and 3) **Logistic Regression** algorithm will be trained with the five selected features.

Creating Different Machine Learning Models

The **first model** that will be built is a supervised machine learning algorithm called **Support Vector Machine**. This model is normally used to classify data points in a database by mapping them to a high-dimensional feature space (**ROADCOND**, **WEATHER**, **LIGHTCOND**, **ADDRTYPE** and **Clus_Db** in this project).

The **second model** is also a supervised machine learning algorithm called **Decision Tree**. This model is also commonly used in classification problems. The main objective of the algorithm is to predict the value of the target variable (**severity** in our project) based on the decisions made using several independent variables (**ROADCOND**, **WEATHER**, **LIGHTCOND**, **ADDRTYPE** and **Clus_Db** in this case).

The **third model** is the last supervised machine learning algorithm that will be used and it is known as **Logistic Regression**. Being a variation of linear regression, Logistic Regression is also useful for classification of categorical dependent variables (**severity**: "**human injury**" or "**property damage**"). Thanks to a sigmoid function, a Logictic Regression model affords the probability of having a severe car accident as a function of all the selected features (**ROADCOND**, **WEATHER**, **LIGHTCOND**, **ADDRTYPE** and **Clus_Db**).

Once the **models** are **trained** with the **train set**, they need to be **evaluated** with the **test set**.

Evaluation of the Machine Learning Models

In order to **evaluate the performance** of the models used different functions will be used **f1_score**, **accuracy_score** and **log_loss** (the latter one only available for the **Logistic Regression** model).

The **F1 Score** is a "weighted average" of the precision and recall, being 1 the best possible value and 0 the worst. It is computed as follows:

$$F1 = 2 \cdot (precision \cdot recall) / (precision + recall)$$

The **accuracy_score** function returns the fraction of correctly "classified" entries (as a float). Thus, the best possible value is also 1 and the worst is 0.

The so-called **Log loss** function estimates the probability of finding an output equal to 1 (in our project severity: "human injury"). This function measures somehow the performance of a classifier being its output a probability.

The following table summarizes all the evaluation results obtained with these two evaluation functions.

Table 1: Results obtained for the different evaluation methods.

ML model	F1-score	Accuracy	LogLoss
SVM	0.5375	0.6697	NA
DT	0.5371	0.6695	NA
LR	0.5372	0.6697	0.6197

As a matter of fact, after training and testing the three different models it can be seen that all of them have a very similar performance. The advantage of the **Logistic Regression** algorithm over the other models used is that it can predict the severity of a car accident but also **its probability** which no other tested model can do.

Training and Testing Models without Geocluster Information

Let's find out whether the position information encoded in the **Clus_Db** column is really useful or on the contrary it is redundant information for the models. In this part of the project, we would like to **demonstrate** that the information contained in the **Clus_Db column is** somehow redundant (or **unnecessary**) since similar information is already encoded in the **ADDRTYPE** attribute.

Once again, a dataframe containing all the attributes required for the new models will be built. The new dataframe will be called **features2** and this time it will only contain the **ROADCOND**, **WEATHER**, **LIGHTCOND** and **ADDRTYPE** attributes (but not the **Clus_Db** attribute).

As done before, the dataset will be splitted into the train and test parts. In order to make the models comparable the dataset needs to be splitted exactly in the same way, *i.e.* the test set containing **24290** entries and the training set containing **137639** entries (15% of the data will be used as a test). Before using the training and test sets they need to be normalized independently. N.B. Before using the training and test sets they need to be normalized independently.

After evaluating all the models (without geocluster information), one can conclude that models perform quite similarly in both cases, with and without geocluster information (Clus_Db feature). Thus, the initial hypothesis formulated in the "**Methodology**" section is confirmed; the geocluster information (geographic position) is redundant and useless information for the models. The following table summarizes all the results obtained.

Table 2: Comparison between results obtained with and without "Geocluster Information".

ML model	F1-score	Accuracy	LogLoss	ML model without Clus_Db	F1-score	Accuracy	LogLoss
SVM	0.5375	0.6697	NA	**SVM**	0.5372	0.6697	NA
DT	0.5371	0.6695	NA	**DT**	0.5371	0.6695	NA
LR	0.5372	0.6697	0.6197	**LR**	0.5372	0.6697	0.6199

Conclusions

In this study case, different supervised machine learning models have been trained in order to predict the **severity of a car accident** in the city of **Seattle**. It has been shown that with the **Logistic Regression** model can also predict the probability of having a severe car accident (when given a set of data points). In order to train the different models, a database containing information about car accidents in the city of Seattle has been used.

The **database** used **contains** all registered **car accidents in Seattle from 2004 to the present**. One might expect that with new improvements in the roads car accidents should have decreased over time, however, the data showed that the decreasing trend is not significant enough not to take into account all the years reported in the dataset.

The three supervised models (**SVM**, **DT** and **LR**) trained with the **ROADCOND**, **WEATHER**, **LIGHTCOND** and **ADDRTYPE** features afford virtually the same evaluation results. **SVM** takes a lot of computation time (it is a **very expensive algorithm**) compared to the other two algorithms, while the **DT** and **LR** are quite cheap. Thus, it is **not worthwhile** using the **SVM model**, since all of the models present similar accuracies.

It has been also shown that the **geographic coordinates** of the car accidents (**encoded in** the geocluster information column: "**Clus_Db**") **do not include** any **valuable information** and thus it is not necessary **to** add them in **the model**. This is due to the fact that the **ADDRTYPE** attribute **already contains** enough **information** about the "**location**" (block or intersection) of the car accident.

The utility of the models developed in this project can be illustrated with an example. Assuming that we can easily interface our model with an application such as Google Maps, and a **driver** (the **stakeholder** of this project) wants to go from south to north (red points in the map below, see **Figure 13**). Google Maps affords **a set of data points** (the green ones for example, see **Figure 13**) **to our developed model** and the **driver knows** for each "step" of the proposed pathway the **likelihood of** having a "**property damage**" or "**human injuries**" car accident. The **LR** model **can** also **predict the probability of having a severe car accident** (with human injuries) **given** the pathway of **data points in the map**. If the initial road does not seem to be safe enough the driver can ask for an alternative pathway less dangerous (the pathway going through the blue points for example, see **Figure 13**).

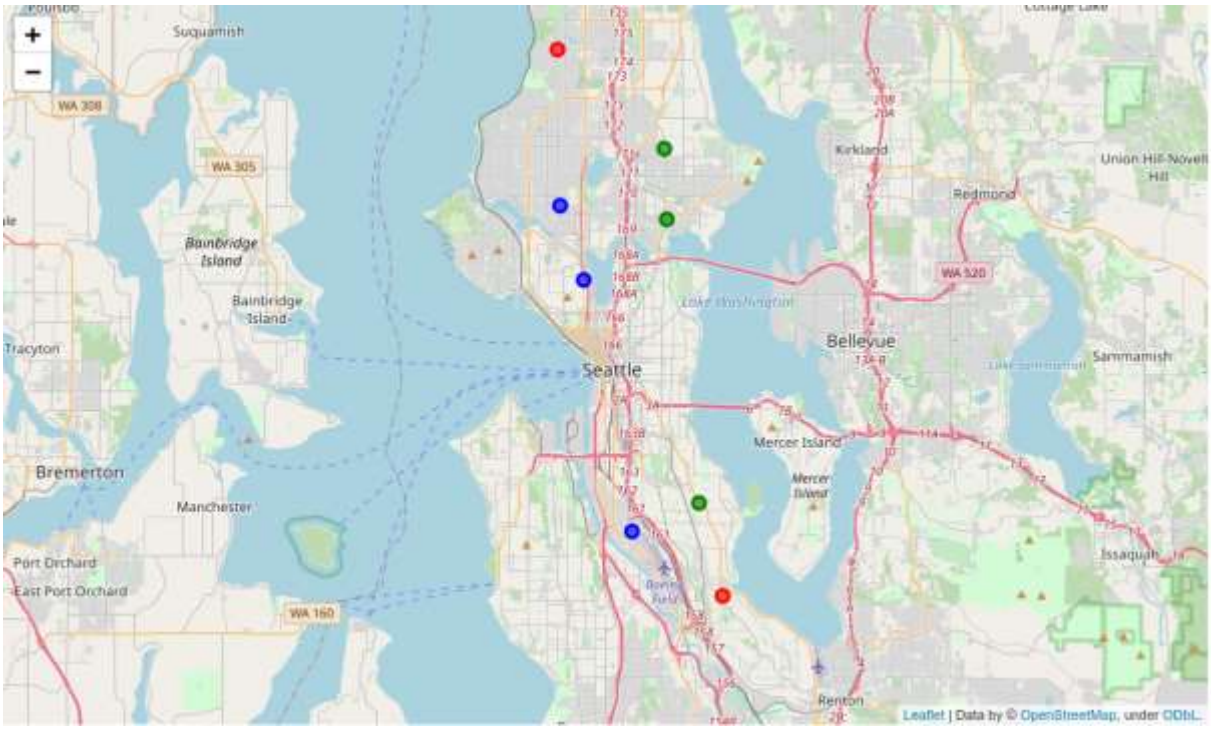

Figure 13: Example of the developed model's utility.

Outlook

In future versions of the model new features such as the time of the accident (**INCDTTM**) could also be included to explore the possible improvements. In order to include the time of the accident maybe the day can be splitted into several parts such as "**early morning**", "**late morning**", **noon**, **afternoon** and **evening**. The daytime may also be splitted more drastically into a **binary data** such as **day** and **night**. Including this information (and maybe some other feature) may be worthwhile exploring in order to analyze the behavior of the predicting models.