Alert Today – Alive Tomorrow: Predicting Road Safety

Sebastiano Denegri

September 05, 2020

I was not predicting the future, I was trying to prevent it.
— Ray Bradbury.

Applied Data Science Capstone – IBM / Coursera.

Table of contents

1. Executive Summary

In this project we predicted the risk level of getting into a car accident, and the severity of a potential collision in the city of Seattle.

Target market:

- *The drivers*: people interested in knowing the probability of getting into a car accident and how severe it would be.
- *Public authorities* to be better prepared in dealing with such events knowing the risk level and the scale of the problem.

Predictive classification models:

- **Model 1 - Classification of Risk**. The model will return a binary-class prediction:
    1. Low risk: less than the average (number of collisions).
    2. High risk: greater or equal to the average (number of collisions).

- **Model 2 - Classification of Collision Severity**. Binary-class prediction:
    1. Not severe (property damaged only);
    2. Severe (injuries and/or fatalities).

Database has been cleaned and processed. Attributes which were either not relevant, redundant, or not available at the time of attempting a prediction were dropped.

Model 1 - Classification of Risk:

K-Means Clustering (Clustering algorithm) + K-Nearest-Neighbors (Classification; K = 8). Performance measured through Cross Validation out-of-sample testing. Metrics used: Accuracy Score, F1-Score.

```
Model 1 - "Classification of Risk" average F1-score: 0.9
Model 1 - "Classification of Risk" average accuracy: 0.9
```

Model 2 - Classification of Collision Severity:

K-Nearest-Neighbors (Classification; K = 8). Performance measured through Cross Validation out-of-sample testing. Metrics used: Accuracy Score, F1-Score.

```
Model 2 - "Classification of Collision Severity" – average F1-score: 0.62
Model 2 - "Classification of Collision Severity" – average accuracy: 0.65
```

Built models can be very useful in warning people about a potential dangerous travel decision.

2. Introduction: The project

2.1. Scenario

“Tomorrow evening I have to go visit my brother's family. He lives in another neighborhood, on the other side of town. Shall I drive there or take the metro? Driving is certainly more convenient; will I run into an accident if I take the car? Shall I move the visit to another day?”

This is a typical everyday situation. To drive or not to drive? Take the car or find another way? If only someone could know. Answering this question is the scope of this project: provide people with a predictive tool, able to warn them about the likelihood of car accidents and their severity, allowing them to make a more informed decision regarding their future travel plans, and choice of transportation.

2.2 Scope of the project

In this project we predicted the risk level of getting into a car accident, and the severity of a potential collision in the city of Seattle, based on historical data collected by the Seattle Police Department (SPD), and the Seattle Department of Transportation (SDOT), from 2004.

The **target market** of the project is:

- Anyone with a driving license, and a mean of transport (“the drivers”), interested in knowing the probability of getting into a car accident and how severe it would be, given factors such as weather, time of the year, location, light conditions..., so that they would drive more carefully or even change their travel plans, if possible;

- Public authorities such as police, departments of transportation, road authorities, healthcare providers... in order to be better prepared in dealing with such events knowing the risk level and the scale of the problem.

Risk class and severity of accidents will be assessed using the available historical data in terms of severity classes, and other available information such as collision locations, date and time of the event, weather, road and light conditions...

In order to deliver a product able to satisfy all relevant stakeholders' needs, I used an analytic, machine-learning driven approach, to build 2 predictive classification models, as follows:

- **Model 1 - Classification of Risk**. Is the risk of getting into car accident higher than usual? The model will return a binary-class prediction, as follows:
    1. Low risk: input feature data match conditions whose “number of collisions” (or frequency distribution) is less than the average number of collisions.
    2. High risk: input feature data match conditions whose “number of collisions” (or frequency distribution) is greater or equal to the average number of collisions.

- **Model 2 - Classification of Collision Severity**. How severe the potential collision is. The model will return a binary-class prediction, as follows:
    1. Not severe (property damaged only);
    2. Severe (injuries and/or fatalities).

3. Data

3.1 Data Requirements

In order to build a predictive model able to forecast the risk and severity of vehicles' collisions, I needed historical data, as more comprehensive as possible, on road accidents in the region/city of interest, in this case Seattle. For better predictive results, the dataset had to include attributes such as:

- Time of the year
- Day of the week
- Time of the day
- Incident's location
- Weather condition
- Light condition
- Road condition
- The collision severity classification

It is important to keep in mind that the selected analytic approach, in this project, was predictive, and not explanatory or descriptive.

> *In predictive modeling [...] criteria for choosing predictors are quality of the association between the predictors and the response, data quality, and* ***availability of the predictors at the time of prediction****, known as ex-ante availability. In terms of ex-ante availability, whereas chronological precedence of X to Y is necessary in causal models, in predictive models not only must X precede Y , but* ***X must be available at the time of prediction****.*
>
> — Prof. Galit Shmueli, Institute of Service Science, College of Technology Management, National Tsing Hua University. Reference: To Explain or to Predict?

Information such as the type of collision, the number of people involved, the kind of reported damage... can be very insightful, and have great causal meaning when it comes to explain the severity of an accident, but can't really be used to build a predictive model because these pieces of information are available only once an incident has already actually occurred (ex: how can someone know the kind of accident, or the number of people involved, when trying to predict in advance the likelihood and the severity of the accident itself?).

Therefore, to build the model's feature set, I selected only the attributes that can be actually used as input for the predictive model.

3.2. Data Source

For the scope of this project I used a dataset containing information about all types of collisions that had happened in the city of Seattle, collected by Seattle Police Department (SPD) and recorded by Traffic Records.

3.3. Data Understanding

The dataset has been built with data on all types of collisions from January 1st, 2004, to May 20th, 2020, containing **194,673 records of collisions, with 37 attributes** (severity label column not included). The database contained a mix of data types, including integers, floats, and text type.

136,485 collisions (70% of total) belonged to the Severity Class 1 (Property Damage Only); since the dataset was quite imbalanced, I used some resampling techniques during the stage of model development and testing.

4. Methodology

In order to build 2 different Classification models (Classification of Risk and Classification of Severity), I had to build 2 different datasets, with different feature sets. I had applied certain criteria for data cleaning and feature selection on both models, whilst some other criteria were relevant for either one of the two models.

I followed the below steps, in order to prepare the needed data/feature sets, and develop efficient Classification Models:

- Data Cleaning: searching for missing values, duplicates, wrong entries…
- Data preparation: feature selection, and data pre-processing. Features were selected/discarded according to criteria such as: availability at the time of prediction, relevancy, redundancy…
- Data Processing: feature/data engineering, data normalization…
- Model Development: different classification algorithms were trained, and tested, with different parameters.
- Model Testing: to evaluate model's performance I used a mix of metrics such as: Jaccard Similarity Coefficient (Accuracy Score), Confusion Matrix, F1-score…. Resampling techniques were also used during the stage of model development and testing.

4.1. Data Cleaning

I performed some Exploratory Data Analysis (descriptive statistics, attribute's unique values, frequency distribution…), checking for duplicates, anomalies, missing data, attribute's relevancy…

The database had a small number of missing values (5,334; 2.74% of the whole dataset) in the X and Y columns (that is the geo-coordinates (longitude and latitude) of the collisions). Since the amount of these missing values is small, I decided to drop these entries.

I examined some anomalous data such as incidents with 0 people involved and incidents with 0 vehicles involved (it seemed odd that a car/vehicle collision didn't involve at least 1 person (how about the driver?) and 1 vehicle). Number of incidents recorded as either involving 0 people or vehicles was less than 3% of whole dataset; after performing some analysis, it was clear that these incidents actually did involve at least 1 person/vehicle. It was likely an error in the attribute, however, considering the small size of these anomalies, I decided to discard them, in order to avoid building a model with erroneous, wrongly recorded entries.

I dropped the duplicates with the same REPORTNO key (only 3 entries).

I dropped the attributes which were either not relevant to the scope of the project (SEVERITYDESC, OBJECTID, INCKEY, COLDETKEY, SEVERITYCODE.1, SDOT_COLCODE, SDOTCOLNUM, REPORTNO, STATUS, INATTENTIONIND, ST_COLCODE, ST_COLDESC, EXCEPTRSNCODE, EXCEPTRSNDESC), redundant (ADDRTYPE, INTKEY, SEGLANEKEY, CROSSWALKKEY, LOCATION, JUNCTIONTYPE), or not available at the time of attempting a prediction (PERSONCOUNT, PEDCOUNT, PEDCYLCOUNT, VEHCOUNT, PEDROWNOTGRNT, SPEEDING, HITPARKEDCAR, COLLISIONTYPE, SDOT_COLDESC, UNDERINFL).

4.2. Data Preparation: pre-processing and feature selection.

After the cleaning, the database contained 178,965 records of collisions, with 7 attributes, (severity label column not included), a mix of data types (integers, floats, and text type) and the same severity class imbalance (70% VS 30%). I grouped the attributes, for further analysis/processing, in 3 clusters:

- **Geo-Information/Location attributes**: X and Y (longitude and latitude of the incident location)
- **Date and Time attributes:** INCDATE (date of the incident), INCDTTM (date and time of the incident)
- **Environmental attributes**: WEATHER (description of the weather conditions - 11 categories), ROADCOND (condition of the road during the collision - 9 categories), LIGHTCOND (light conditions during the collision - 9 categories).

4.2.1. Data pre-processing.

I checked for redundancy between the attributes "WEATHER" and "ROADCOND".

```
WEATHER attribute, "Clear" category - ROADCOND distribution (normalized va
lues):
Dry               0.955077
Wet               0.032816
Ice               0.005830
Unknown           0.004391
Snow/Slush        0.000876
Other             0.000562
Sand/Mud/Dirt     0.000248
Oil               0.000152
Standing Water    0.000048
Name: ROADCOND, dtype: float64

WEATHER attribute, "Raining" category - ROADCOND distribution (normalized
values):
Wet               0.972951
Dry               0.018844
Snow/Slush        0.002916
Standing Water    0.002500
Unknown           0.001058
Oil               0.000801
Ice               0.000449
Sand/Mud/Dirt     0.000320
Other             0.000160
Name: ROADCOND, dtype: float64

WEATHER attribute, "Overcast" category - ROADCOND distribution (normalized
values):
Dry               0.582883
Wet               0.389583
Unknown           0.012350
```

```
Ice                0.008347
Snow/Slush         0.004834
Other              0.000906
Sand/Mud/Dirt      0.000567
Oil                0.000302
Standing Water     0.000227
Name: ROADCOND, dtype: float64
```

It was clear there was a huge redundancy between the "WEATHER" categories "Clear" and "Raining" and the "ROADCOND" categories "Dry" and "Wet". The "WEATHER" "Overcast" category didn't really overlap with the "ROADCOND" attribute categories though. Without going any further with other statistical analysis, I considered the "availability at the time of prediction" principle, so I decided to keep only the "WEATHER" attribute, to avoid redundancy, and because predicting what the Road Conditions might be during an "overcast" day, for example, is simply not feasible.

I, then, applied few changes as follows:

- Categorized the missing values in the attributes WEATHER and LIGHTCOND as "Unknown".
- In the "LIGHTCOND" attribute, I grouped the "Dark - No Street Lights" and "Dark - Street Lights Off" categories.

Date and Time attributes. After converting the object date/time attributes to datetime object, I created "month", "day_of_the_week", "hour_of_day", "day_period", and "weekend" columns.

For the "day_period" attribute, I grouped the 24 hours into 4 categories:

- 1: night -> from midnight to 6 am
- 2: morning -> from 6 am to 12 pm
- 3: afternoon -> from 12 pm to 6 pm
- 4: evening -> from 6 pm to midnight

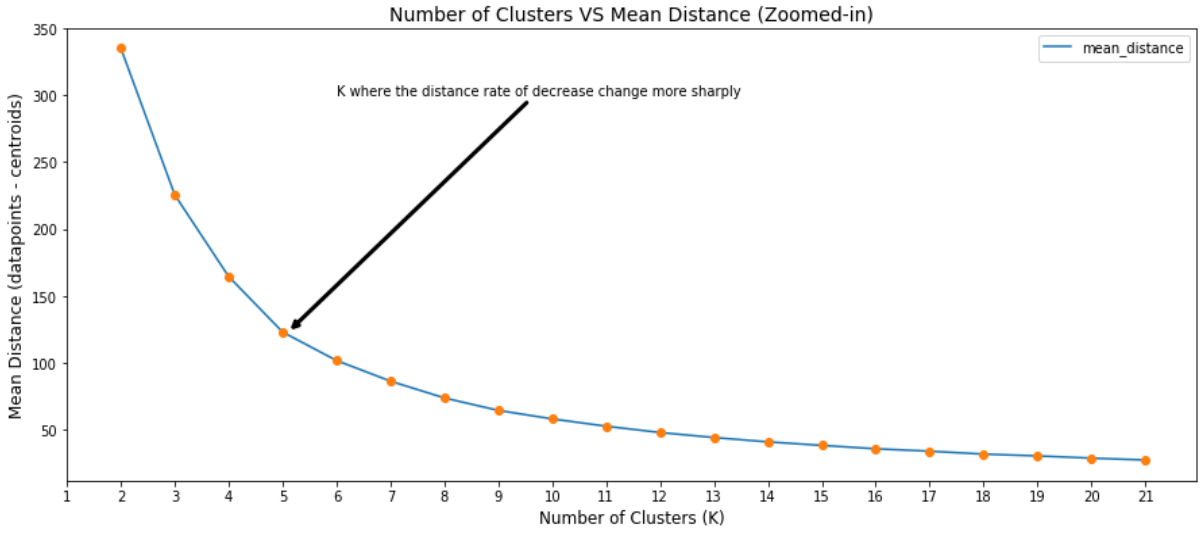
The "weekend" attribute has 2 values:

- 0 -> No
- 1 -> Yes

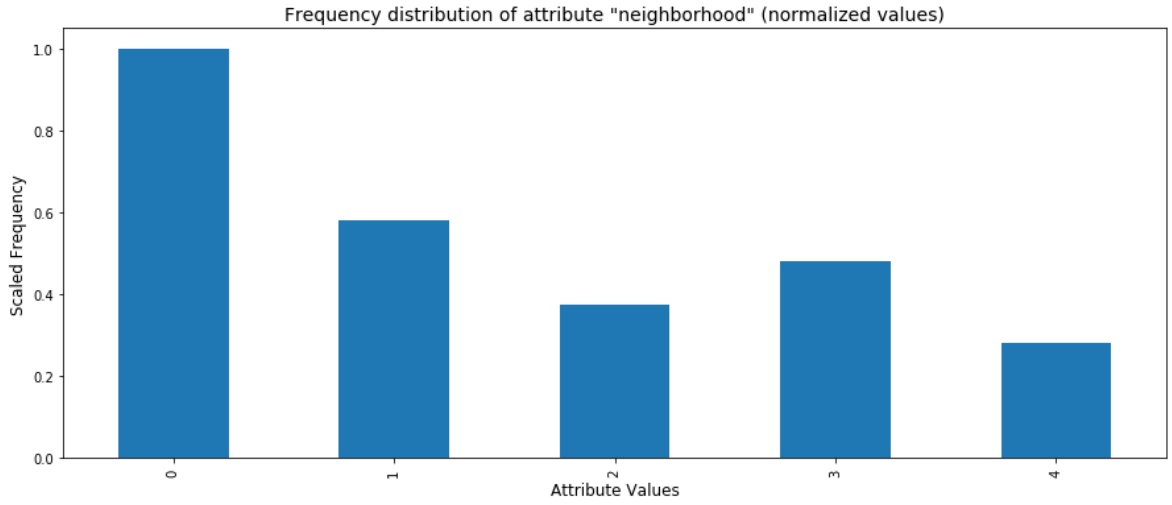
4.2.2. Model 1 - Classification of Risk.

Geo-Information/Location attributes. In order to develop the Model 1 - Classification of Risk, I had to build a new dataset from the SDOT historical data.

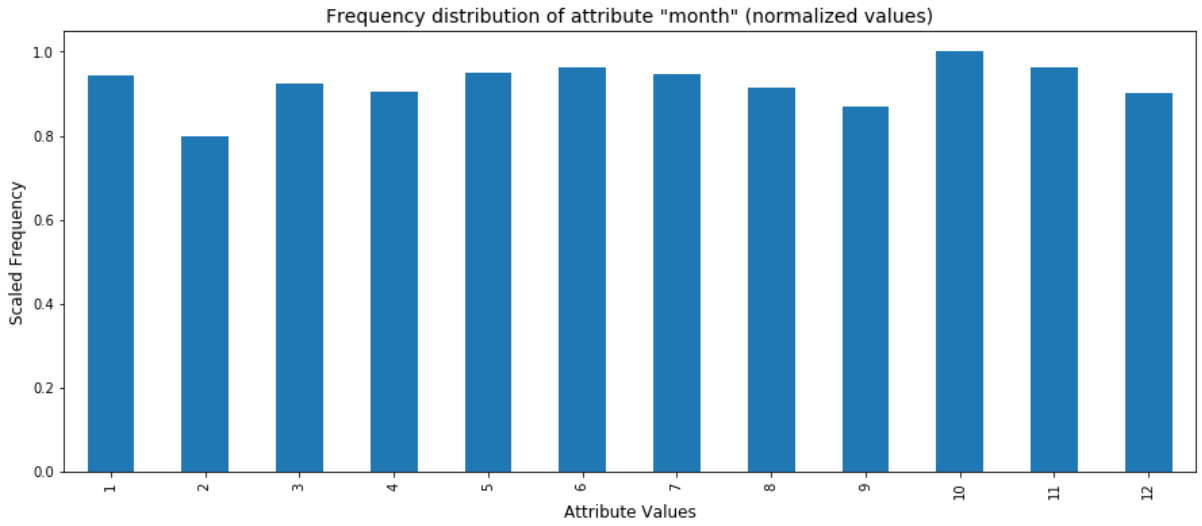
To prepare this dataset, I assigned the collision's geo coordinates (latitude and longitude) to different clusters (for Reverse Geocoding, see Appendix). After collecting and standardizing the X/Y data with StandardScaler, I used the partition based clustering algorithm K-Means. Since with K-Means I had to pre-set the number of clusters (k), I ran the algorithm with different k values (from 2 to 50), measured the mean Euclidean distance (datapoints - centroids) for each iteration, and then compared the mean distance per each k. See following graph:

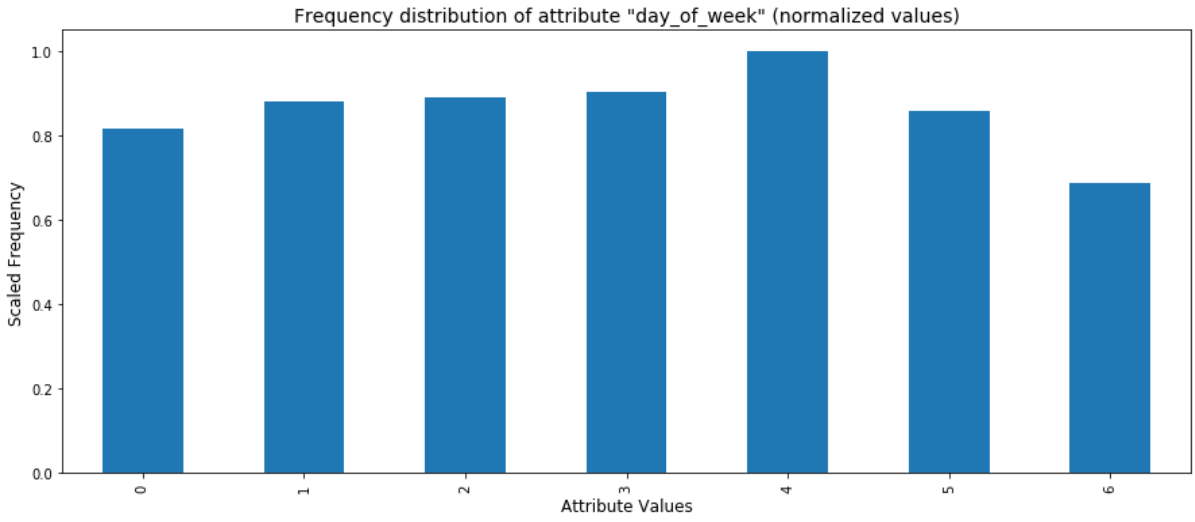


Following the “Elbow rule”, I set the number of clusters as 5, assigned each collision to its geo-cluster ("neighborhood"), and checked the Frequency distribution of the attribute “Neighborhood” (see below).

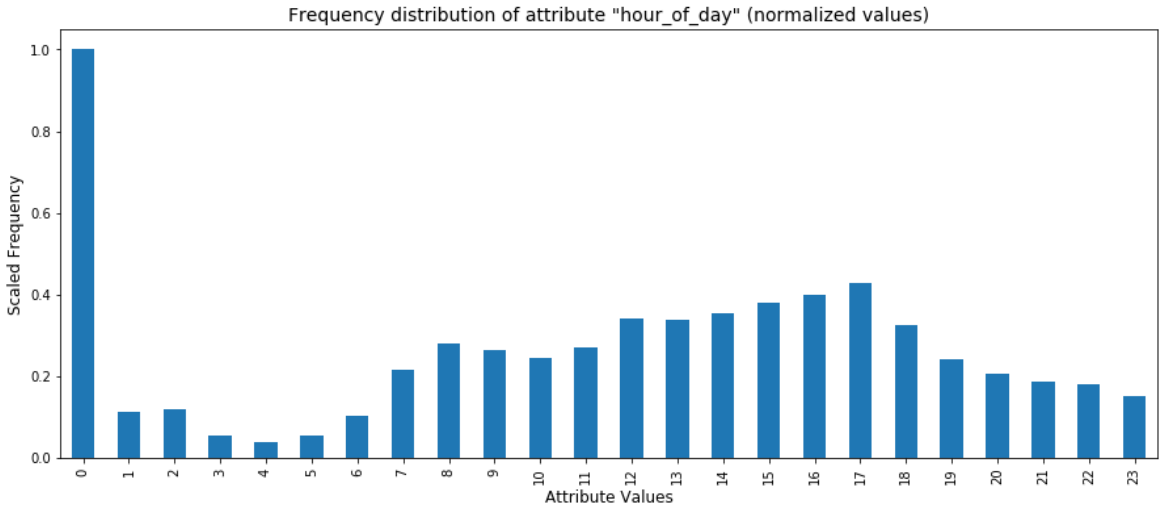


Date and Time attributes. As it shows from the plots below, the number of collisions is actually quite equally distributed over the 12 months and 7 days of week.





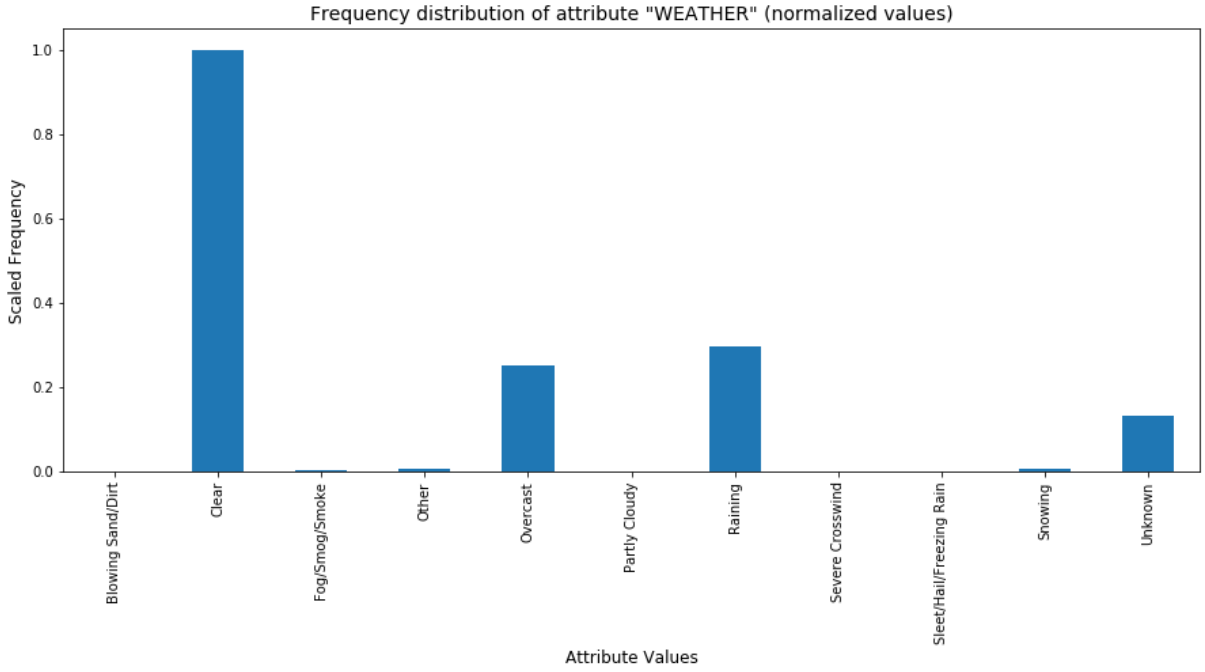
The collisions distribution over the 24 hours is more interesting for the project scope. The midnight hour experiences a way higher number of collisions compared to all other hours of the day, whilst the other “night hours” have a way lower frequency (see below).



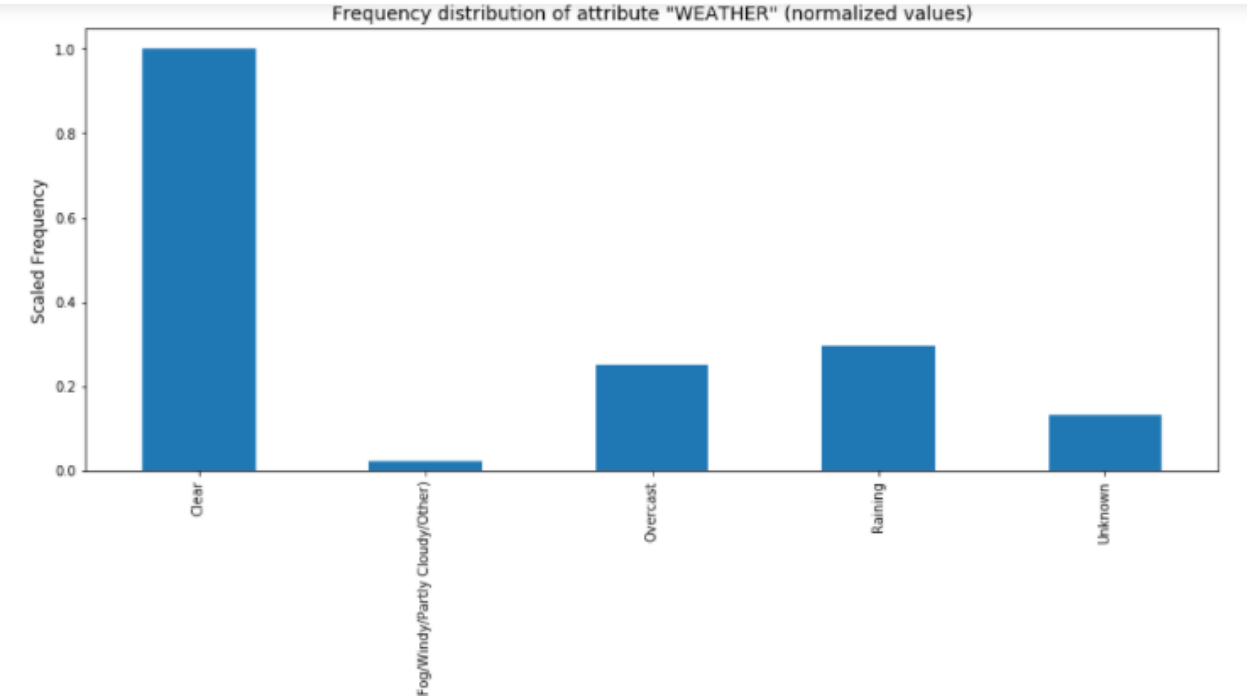
The hour_of_day variable has a way greater impact on the number of collisions than any other date/time attribute.

Environmental attributes. After checking the frequency distribution of the "WEATHER" and “LIGHTCOND” attributes, I grouped some of the attributes’ categories together, in order to reduce the noise.

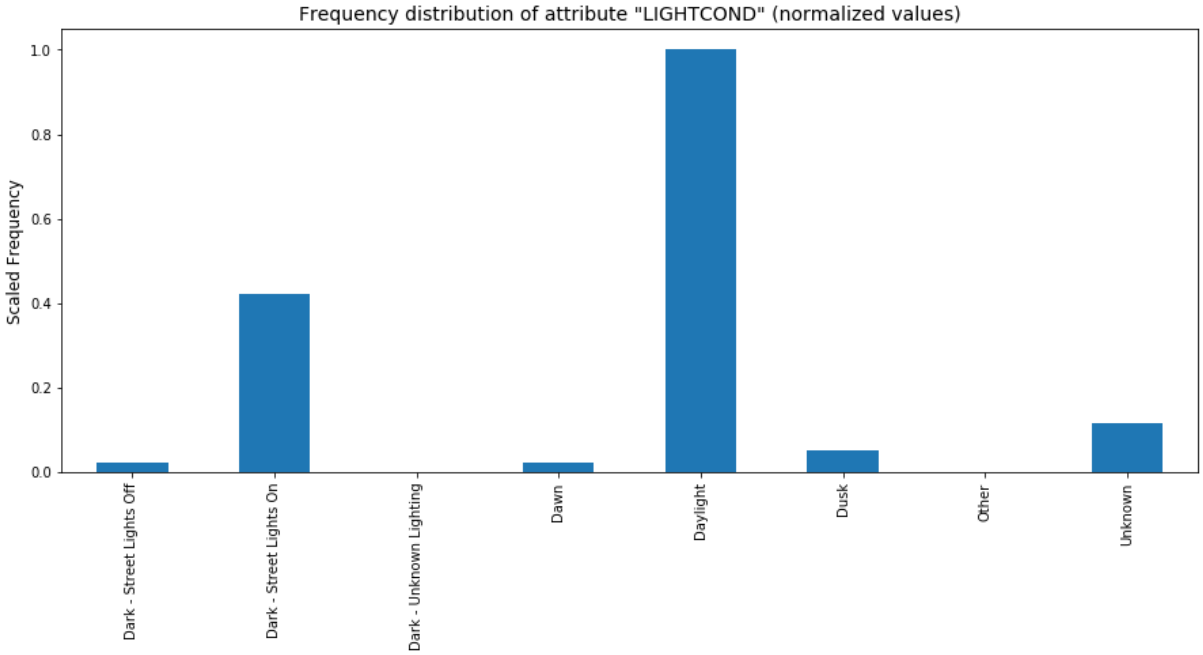
WEATHER attribute categories – before grouping:



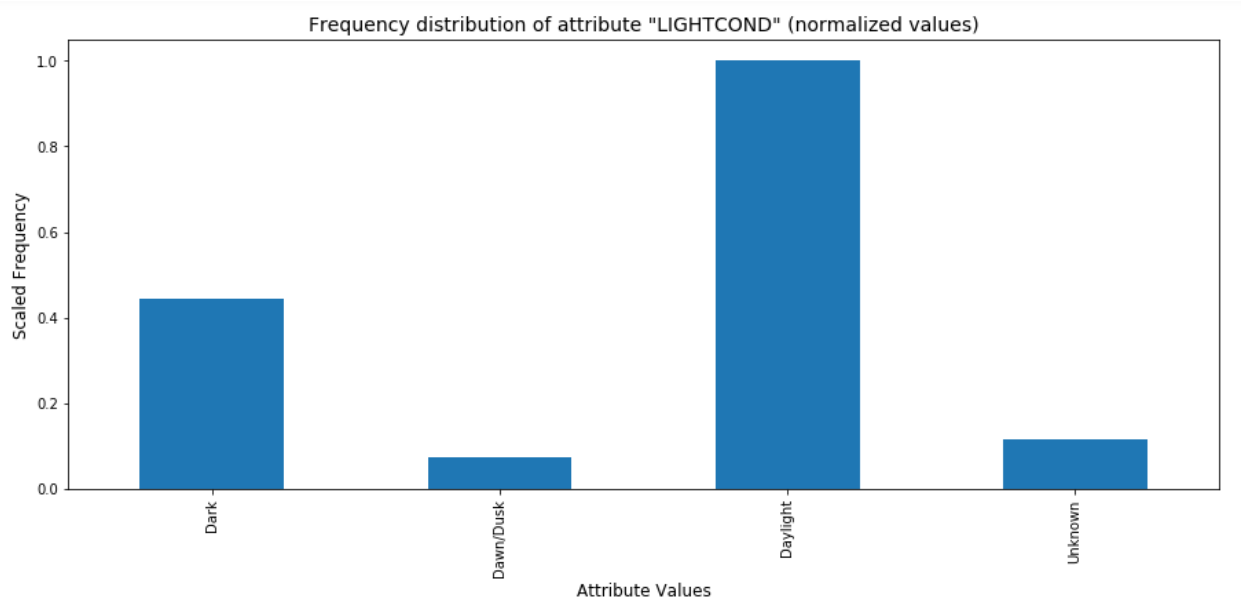
WEATHER attribute categories – after grouping:



LIGHTCOND attribute categories – before grouping:



LIGHTCOND attribute categories – after grouping:



Feature selection. The following are the select features for "Model 1 - Classification of Risk": Neighborhood, hour_of_day, Light Conditions, Weather. To build the dataset, I applied "groupby" method on the select attributes, computed the frequency (= Number of collisions), and assigned the Risk Classes as defined in the scope of the project:

- Class 1: Low risk (less than the average number of collisions).
- Class 2: High risk (greater or equal to the average number of collisions).

	neighborhood	hour_of_day	light_conditions	weather	number_of_collisions	mean	risk_codes
0	0	0	Dark	Clear	2180	91.589048	2
1	0	0	Dark	Other (Snow/Sleet/Hail/Fog/Windy/Partly Cloudy...	59	91.589048	1
2	0	0	Dark	Overcast	587	91.589048	2
3	0	0	Dark	Raining	904	91.589048	2
4	0	0	Dark	Unknown	115	91.589048	2

4.2.3. Model 2 - Classification of Collision Severity.

Date and Time attributes. I used ANOVA (Analysis of Variance) technique to calculate the different F-test scores for different attributes, using as target variable the SEVERITYCODE classification, and compared them to check which attribute has the highest score, that is the highest variation between different category means (over the variation within the category groups). The approach of using ANOVA for a categorical target variable (that is not a continuous one) may be unorthdox, but the purpose here was not to find a correlation coefficient between variables, but comparing the variation of different attribute categories to check whether a

particular attribute had a higher impact over the Severity Classification, and verify if we could discard redundant/irrelevant attributes.

```
Dictionary with attribute name and relevant F-test score:
{'month': 9.08747983999812,
'day_of_week': 13.639657854282587,
'hour_of_day': 31.840191001236775,
'day_period': 103.00228194421788,
'weekend': 56.23912077622821}
```

The F-test scores showed that the categorization of the daily 24 hours in 4 day periods ("1: night", "2: morning", "3: afternoon", "4: evening") had, by far, the biggest impact on the Collision-Severity classification (a higher distance between category means (and a lower variance within categories) indicates a kind of categorization more capable to explain the target variable because more impactful in establishing the target classes). The attribute Weekend was the second most impactful attribute amongst the 5. The "month" attribute was the least impactful.

Environmental attributes. After verifying that “WEATHER” and “LIGHTCOND” were relevant attributes for the Collision-Severity Classification, I performed the Analysis of Variance (ANOVA), to check which one of the different attribute's categorization (“before grouping” VS “after grouping”) was more impactful for predicting the severity classes. The new categorization (“after grouping”) for both the attributes had greater F-Test scores (WEATHER, old categories: 463.68; new categories: 1142.97; LIGHTCOND, old categories: 637.19, new categories: 1465.72).

Feature selection. The following are the select features for “Model 2 - Classification of Collision Severity”: x (longitude), y (latitude), weather, light_conditions, day_period, weekend.

	severity_classes	x	y	weather	light_conditions	day_period	weekend
0	2	-122.323148	47.703140	Overcast	Daylight	3	0
1	1	-122.347294	47.647172	Raining	Dark	4	0
2	1	-122.334540	47.607871	Overcast	Daylight	2	0
3	1	-122.334803	47.604803	Clear	Daylight	2	1
4	2	-122.306426	47.545739	Raining	Daylight	2	0

4.3 Data Processing

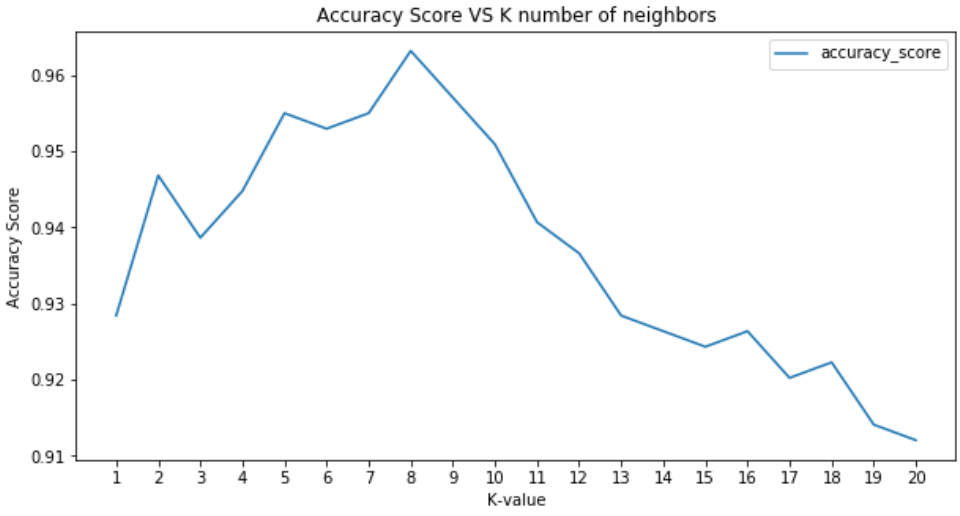
Both feature sets (Model-1 and Model-2) have been processed by turning categorical values into numeric values (get_dummies method), converting the dataframes into Numpy arrays, and standardizing data with Standard Scaler.

4.4. Model Development and Testing

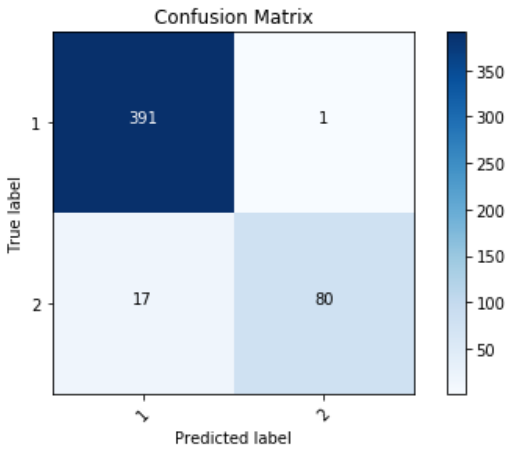
4.4.1. Model 1 - Classification of Risk

For this model, I used K-Nearest-Neighbor (KNN) algorithm. After splitting the dataset into train (75%) and test (25%) sets, I tested the algorithm with several values for K (the number of neighbors), to find the best K value. It turned out that 8 was the K value with the highest accuracy score. See below train and test sets size, and accuracy score VS K line plot.

- Shape of train subset: 1,464 entries
- Shape of test subset: 489 entrires



I ran, then, the KNN algorithm with K value as 8, and performed Confusion Matrix analysis to check the F1 score, and precision/recall for both classes.



	precision	recall	f1-score	support
1	0.96	1.00	0.98	392
2	0.99	0.82	0.90	97
accuracy			0.96	489
macro avg	0.97	0.91	0.94	489
weighted avg	0.96	0.96	0.96	489

The model worked very well. Not only we have obtained a very high accuracy score (96%), but, despite the fact that the dataset is quite imbalanced, we've got a very high F1-score (the harmonic average of precision and recall), which means our model is able to predict cases classified as 2 (higher risk of getting into a car accident) with almost perfect precision (99%) and good recall as well (82%).

4.4.2. Model 2 - Classification of Collision Severity

For the second model, I used Decision Tree, Logistic Regression, and K-Nearest-Neighbors (KNN) algorithms. Considering the size of the dataset (178,965 rows), I decided not to use algorithms that work well with small databases like Support Vector Machine.

After splitting the dataset into train (75%) and test (25%) sets, I tested the different algorithms. Regarding the algorithm's parameters, I set the followings:

- Decision Tree -> Max Depth = None;
- Logistic Regression -> I ran the algorithm with different solvers and regularization parameters, and choose the parameters with the greatest F1-Score: Solver = liblinear; Regularization = 0.1;
- KNN -> I tested the algorithm with several values for K (from 1 to 20), to find the best K value. It turned out that 7 was the K value with the greatest F1 score.

See train and test sets size, and **Report Table-1** below:

- Shape of train subset: 134,223 entries
- Shape of test subset: 44,742 entries

	Algorithm	Accuracy_score	F1-score
0	Decision_Tree	0.628604	0.620514
1	Logistic_regression	0.697331	0.572983
2	KNN	0.652876	0.618652

Logistic Regression performed very poorly on the minority class prediction; therefore, despite the highest accuracy, Logistic Regression was definitely the algorithm that performed worst. Decision Tree and KNN algorithms had very similar F1 scores, but KNN had a slightly greater accuracy, although Decision Tree seemed to perform better on predicting cases belonging to the minority class (see the Confusion Matrices in the notebook). However, the minority class had quite low precision, recall, and F1-Score values for all 3 algorithms. I decided to try, then, some resampling techniques such as:

- Oversample minority class
- Undersample majority class
- Generate synthetic samples (SMOTE)

Report Table 2:

	Dataset	Dec_Tree - F1	Log_regr - F1	KNN - F1	Dec_Tree - Accuracy	Log_regr - Accuracy	KNN - Accuracy
0	No_resample	0.620514	0.572983	0.618652	0.628604	0.697331	0.652876
1	Oversampled_minority	0.606161	0.421178	0.579404	0.611148	0.442917	0.564146
2	Undersampled_majority	0.571530	0.417180	0.563811	0.555474	0.440503	0.545394
3	Synthetic_samples_(SMOTE)	0.555761	0.433354	0.579878	0.536990	0.449980	0.563497

All resampled datasets performed worse than the original one, for each algorithm.

5. Results - Cross Validation

5.1. Model 1 - Classification of Risk

For further testing and validation, I performed Cross Validation, as an additional out-of-sample metrics. We obtained an average F1-Score and Accuracy_Score of 90%, which is very good. The Model 1 – Classification of Risk of getting into a car accident has performed very well:

```
Model 1 - "Classification of Risk" average F1-score: 0.9
```

```
Model 1 - "Classification of Risk" average accuracy score: 0.9
```

5.2. Model 2 – Classification of Collision Severity

Since Decision Tree and KNN algorithm's performance is very close, I performed Cross Validation analysis on both algorithms as a further (out-of-sample) metrics and checked which model works better.

```
Model 2: "Classification of Collision Severity" –
Decision Tree average F1-score: 0.62
Model 2: "Classification of Collision Severity" –
Decision Tree average accuracy score: 0.62
Model 2 - "Classification of Collision Severity" –
KNN average F1-score: 0.62
Model 2 - "Classification of Collision Severity" –
KNN average accuracy score: 0.65
```

Decision Tree worked slightly better with regard of the minority class, but overall KNN had a greater accuracy. See Classification Reports below:

Decision Tree – Classification Report:

```
[[23799  7401]
 [ 9216  4326]]
              precision    recall  f1-score   support

           1       0.72      0.76      0.74     31200
           2       0.37      0.32      0.34     13542

    accuracy                           0.63     44742
   macro avg       0.54      0.54      0.54     44742
weighted avg       0.61      0.63      0.62     44742
```

KNN – Classification Report:

```
[[26438  4762]
 [10769  2773]]
              precision    recall  f1-score   support

           1       0.71      0.85      0.77     31200
           2       0.37      0.20      0.26     13542

    accuracy                           0.65     44742
   macro avg       0.54      0.53      0.52     44742
weighted avg       0.61      0.65      0.62     44742
```

Considering KNN greatest accuracy, I decided to pick KNN (K=7) as the algorithm for Model 2 – Classification of Severity Collision.

6. Discussion

For the first model, I was able to achieve 90% accuracy (with a F1-Score of 0.9), using K-Nearest-Neighbors algorithm (number of neighbors = 8). It is important to note that Model 1 – Classification of Risk doesn't predict the likelihood of getting into a car accident, but it returns a binary classification able to predict the class of risk based on the average number of total collision that took place in the considered timeframe (a high risk means that input conditions, historically, are those ones with a number of accidents greater than the mean). This approach might be considered "naif", however I reckon this kind of model can give a "warning" to a driver who's planning a car-trip under conditions considered dangerous from an objective point of view (that is conditions under which the majority of accidents happen).

For the second model, I achieved 65% of accuracy (with a F1-Score of 0.62), again using K-Nearest-Neighbors algorithm (number of neighbors = 7), therefore there's still room for improvement. Resampling techniques didn't return any better results; I tried the same algorithms with geo-clusters (instead of latitude and longitude), and even different feature selection (choosing only features with ANOVA F-test scores above a certain threshold), but, again, outputs weren't any better (actually even worse). My conclusion is that the severity of a potential collision is a difficult prediction to make; features such as "vehicles speed", "total people involved", ''driver under alcohol/drug influence", might significantly help, but they are not known at the time of attempting a prediction, so they can't be used; however, "individual/per vehicle" features of this type might also contribute to a collision's severity, and, therefore, to the model's performance. For instance, knowing the number of people in each vehicle, during a collision, might be helpful: in this case a potential user knows (or anyway might have an idea of) how many people will be in his/her own vehicle and the feature can be used to attempt a prediction. These types of data are obviously more difficult to obtain, but they could bring significant improvements to the model.

7. Conclusion

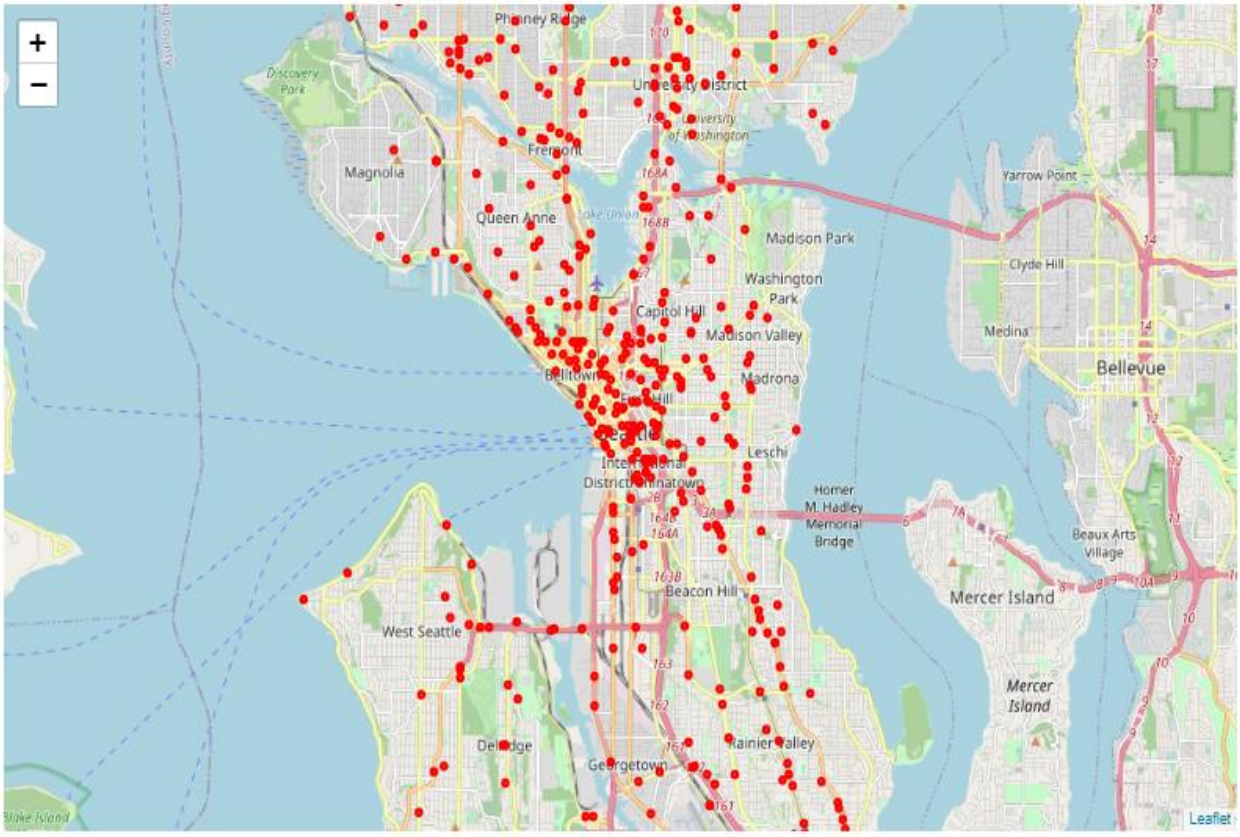
In this project, I built 2 classification models for potential users to enable them making a more informed decisions about their future travel plans. I selected K-Nearest-Neighbors (KNN) as the algorithm for both models. Built classification models predict whether the risk of getting into a car accident is high (that is higher than the mean) or low (Model 1), and whether the severity of a potential collision will be low (only property damaged) or high (people injured/fatality).

The first model (Class of Risk) is actually made of 2 machine learning models: a clustering-algorithms (K-Means), and a classification one. Users will input the destination location of their trip (amongst other input features), the model will cluster the geo-coordinates, and return the Class of Risk.

For both models I identified location, time and environmental (weather and light conditions) features as the most important features that affect the risk of getting into a car accidents, and its severity. The models can be very useful in warning people about a potential dangerous travel decision.

9. Appendix

Incident's locations (random sample of 500 incidents) - Seattle map.



Reverse Geocoding

Another way to cluster collision's coordinates (instead of using a clustering algorithm like K-Means) could have been linking each pair of coordinates (latitude and longitude) to their actual neighborhood and/or postal codes. This is called Reverse Geocoding, which is the process of back (reverse) coding of a point location (latitude, longitude) to a readable address or place name (this permits the identification of nearby street addresses, places, and/or areal subdivisions such as neighborhoods, county, state, or country).

To perform Reverse Geocoding, I should have used **geocoder library**, with an **API key from Google Cloud Platform services**. The problem with this approach is that there is an API limit of use, and it takes very long time to process all the rows in the dataset, so, for the scope of this project, I've performed a clustering algorithm instead.

For Reverse Geocoding code, please, see the following:

```python
# import geocoder for reverse geocoding
import geocoder
# Import progressbar to measure the progress of the whole process
from tqdm import tqdm

# Convert the attributes to tuples to speed up the process
latitude = tuple(data_collisions['Y'])
longitude = tuple(data_collisions['X'])
# Create 2 empty lists to be filled with geo information
neighborhoods = []
postal_codes = []

with tqdm(position=0, leave=True) as pbar:
    for lat, long in tqdm(zip(latitude, longitude), position=0, leave=True):
        location_info = geocoder.google([lat, long], method='reverse', key=api_key)
    # check if the field "neighborhood" is in the dictionary returned by geocoder; if not, returns a null value
        if 'neighborhood' in location_info.json.keys():
            neighborhoods.append(location_info.json['neighborhood'])
        else:
            neighborhoods.append(0)
    # check if the field "postal" is in the dictionary returned by geocoder; if not, returns a null value
        if 'postal' in location_info.json.keys():
            postal_codes.append(location_info.json['postal'])
        else:
            postal_codes.append(0)

print('List of first 100 collision neighborhoods:', neighborhoods[0:100])
print('List of first 100 collision postal codes:', postal_codes[0:100])
```

Reference Notebook link:

https://nbviewer.jupyter.org/github/SebastianoDenegri/Coursera_Capstone/blob/master/applied_data_science_capstone-final_version_geocoord.ipynb