Road Accidents in France

Description:

The objective as specified by the DS[1] team was to predict the severity of Road Accidents in France based on historical data[2]. A zone scoring system was then to be created to categorize or score the zones of France according to the risk of road accidents.

The analysis was done from the years 2019-2023 and the area was limited to mainland France

Goals:

- Model a ML/DL[3] model that can either predict the severity of the accidents.
- Or create a Forecaster that can predict future accidents.
- Create a DangerZone Scoring System
- Visualize the Danger Zones

[1] DataScientest
[2] https://www.data.gouv.fr/fr/datasets/53698f4ca3a729239d2036df/
[3] Machine Learning/Deep Learning hence forth as MDL

Road Accidents in France

Report 1: exploration, data visualization and data pre-processing report

Introduction to the project

Context:

The project and the data sources consist of analyzing the road accidents that happened in France from the year 2005 to current year. The idea is to build a classifier to predict the severity of the road accidents.

Road Accidents are the leading cause of death worldwide for children and young adults till aged 30 causing over 1 million deaths annually (WHO).
This heavy cost not only incurs a toll on the future of the country but costs an estimated 3% loss of GDP to most countries.

These critical losses can be thus analyzed and addressed by modern MDL[4] techniques. Advancement in the domain of MDL would allow us to understand the nature of and find patterns to curtail the growing number of loss of life due to traffic accidents. MDL would allow researchers to then suggest preventive or protective measures to decrease the number of fatalities.

Objectives:

The goals of the project broadly ask the question whether we can find such patterns and can MDL help us recognize zones of France[5] associated with the most risk due to road accidents.

For this project we decided to use mostly mainland France and the years 2019 and beyond.

We thus list out our main goals for each step of the project as follow:

- Analyze the data and validate its purpose (Basically understand what the data entails) and convert into a human readable english form[6].
- Filter and clean the data (Preprocessing)
- Model the data to predict the severity
- Calculate and visualize the Danger Zones

---

[4] MDL : Machine Learning / Deep Learning
[5] Danger Zones
[6] No translations were provided till almost 20 days after the project started and only 1 subset

The description of the criteria of the project asked for that we have expertise in the data analyzed. Apart from being a traveller and a regular user of vehicles , I possess no expertise in road-accidents or traffic regulations.

The over 100's of reuses and visualizations[7] shows a lack of forethought into selecting the data source by the DS team. Nevertheless data being data , patterns can be quickly analyzed and assessed and models can be built. Road Accident analysis is quite common and there are quite a few headways into the research and designing of MDL models to create traffic policies to decrease the ill effects of road accidents. (Behboudi)

Understanding and manipulation of data

Framework:

- We selected road accident data from the years 2019-2023 (Ministry of Interior)
- The data was with an Open License and thus was free to use.
- The data was over 200MB in an uncompressed csv format and consisted of over 500k values [8].
- The data was also split into 4 subsets called (*users, vehicles, characteristics, places)* which was then merged to make it easier to deal with.

Relevance:

The data was in French and encoded with short form(see Figure 1) was quite difficult to decipher as the keys were outdated and not properly provided in english. Most time in the beginning was spent in deciphering what these codes stand for. Once deciphered we realized that the variable that was significant was *grav* which indicated the severity of the accident.

Our first task was to convert the understandable and translated variables into a human readable form. We achieved that by creating a custom *Enum* class that held the values of the coded variables (see Figure 2 and notebook *Transformer.ipynb*).

catv	obs	obsm
2	0	2
7	0	2

(Figure 1: French Sample of variables)

[7] https://www.data.gouv.fr/fr/datasets/53698f4ca3a729239d2036df/#/community-reuses
[8] A mid to small data size

```
    class traffic(CategoryBaseEnum):
        '''Traffic regime:'''
        NOT_PROVIDED = -1
        ONE_WAY = 1
        BIDIRECTIONAL = 2
        SEPARATE_LANES = 3
        VARIABLE_LANE = 4


    class road_surface(CategoryBaseEnum):
        '''Surface condition'''
        NOT_PROVIDED = -1
        NORMAL = 1
        WET = 2
        PUDDLES = 3
        FLOODED = 4
        SNOWY = 5
        MUDDY = 6
        ICY = 7
        GREASE = 8
        OTHER = 9
```

(Figure 2 : Sample Converted Variables from the Enum Class)

The translations helped us decide the columns which made the most sense that would influence the target variable of *severity*.
The dataset was of course converted to english variables with decipherable names which would make it easier to perform further *EDA*[9] and other visualizations.

```
class lum(CategoryBaseEnum):
    '''Lighting conditions during the accident'''
    UNKNOWN = 0
    DAYLIGHT = 1
    NIGHT_LIT = 2
    LOW_LIGHT = 3
    NIGHT_DARK = 4

class weather(CategoryBaseEnum):
    '''Weather conditions during the accident'''
    UNKNOWN = 0
    LIGHT = 1
    MEDIUM = 2
    SEVERE = 3

class collision_type(CategoryBaseEnum):
    '''Type of collision'''
    UNKNOWN = 0
    NO_COLLISION = 1
    SIMPLE = 2
    COMPLEX = 3
```

(Figure 3 : Simplifying the variables via Hierarchy)[10]

Pre-processing and feature engineering:

The translations helped dealing with which features could be selected based on common sense [11] . The data being merged had over 40 variables and over 500k values and after the common preprocessing and data cleaning techniques we were able to get rid of unclean data. Normally for a smaller data set if variable values were missing an Imputation would be advised , however since we had a

[9] EDA : Exploratory Data Analysis.
[10] This simplification allows for a nice way to define ordinality
[11] For e.g. *weather* plays a part in road accidents.

relatively large dataset it was acceptable to clean and filter out any values that had missing variables.(See *Preprocessing.ipynb & EDA-France-Road-Accident.ipynb & monolith.py for code)*

The next task was to filter out variables with unusable values[12] or limit the values. We hardcoded the limits of France and removed any values not in this limit . See code snippet below

```
##lat lon cleaning # Only Keep mainland france and the tiny island nearby
mask = (data['lat'] >= LAT_MIN) & (data['lat'] <= LAT_MAX) & (data['long'] >= LONG_MIN) & (data['long'] <= LONG_MAX)
data = data[mask]
```

The next big decision was how to use the GPS[13] coordinates which were present in the form of *lat & long*. It made sense rather than using the singular coordinates but to cluster the accidents to visualize clusters of accidents. In the next step we specify how to encode the GPS data.

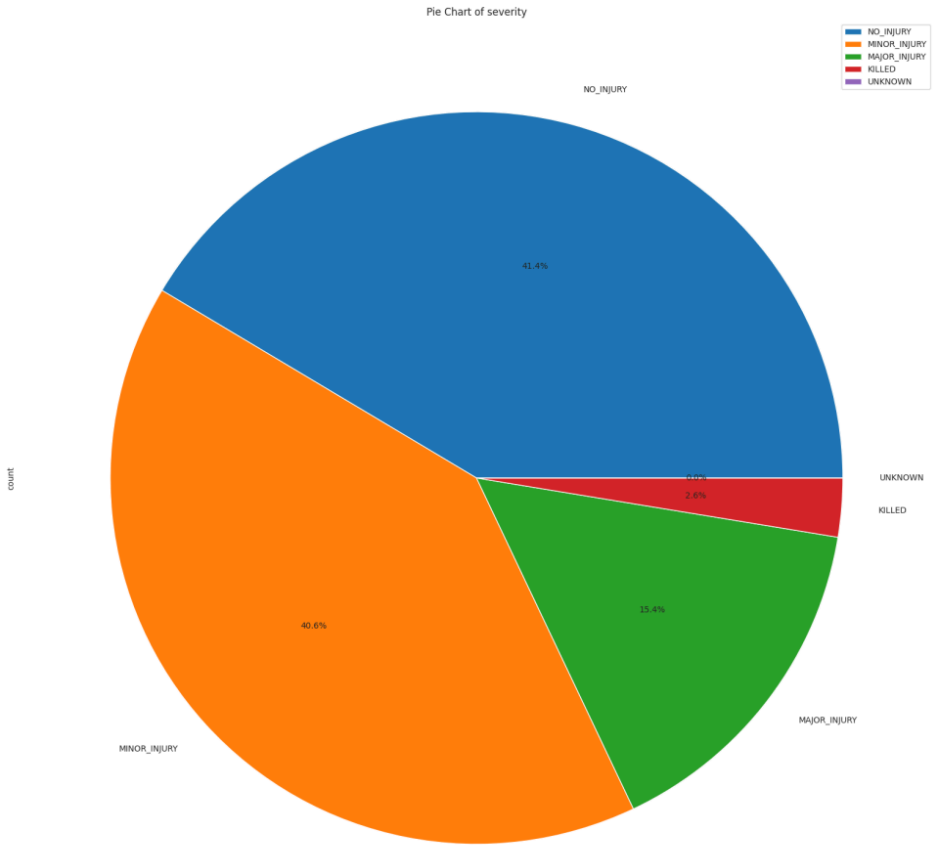
Normalization/Standardization was generally not needed. Some *numerical* variables we converted to a proper scale but in large since most of the variables belonged to a category we did not have to do much.

Common sense was a first dimensionality reduction technique in which we identified about 10 or so variables influencing the road accidents. The *Enum class* called *RoadAccidentEnum* allowed us to keep a tally of the variables needed and was a central store of the key variables and their corresponding values. This also held a check to see if the variable was a categorical one which allowed us to filter out categorical variables.

[12] speed_limit : national speed limit was 140. age : < 100
[13] GPS: Global Position System coordinates in the form lat(Latitude) and long(Longitude)

Visualization & Statistics :

The dataset was quite imbalanced especially the target variable *severity* (see Figure 4) and in this case it would be hard to predict the extreme fatalities without sampling methods that address the imbalance[14].



(Figure 4 : Severity Pie Chart)

This imbalance was a theme throughout and would prove problematic. A small example was the *lum* (see Figure 5) which noted the luminosity at the time of the accident . As we can see most accidents noted in the period occurred during the day time.

For columns that had greater categories[15] , the imbalance was quite evident and it made sense to group these categories together and create major categories to help create manageable variables. This idea of simplification for the variables led us to create a cluster of categories into manageable formats. This allowed us to assign a hierarchical system which dictated the ordinality of categories. (See Figure 6 for examples). Here *road_surface* has been clustered into manageable categories and its values represent the impact on accidents.[16]

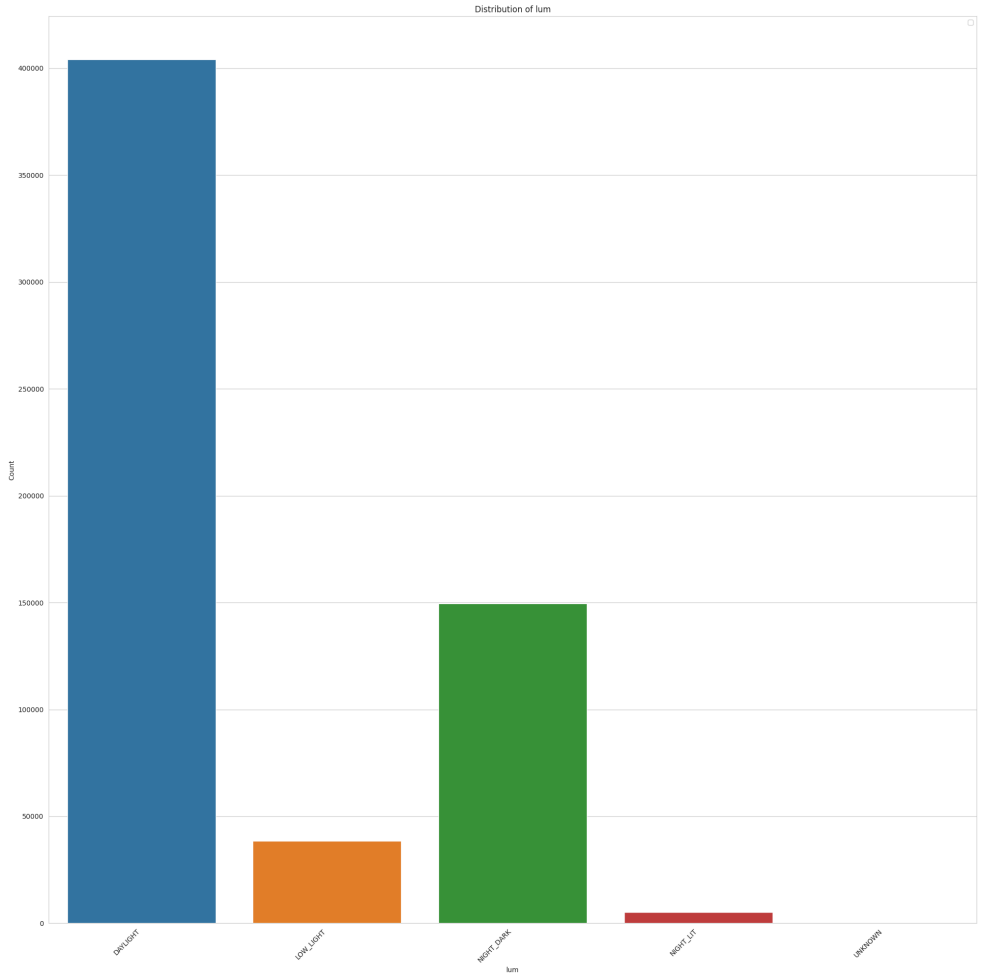
[14] SMOTE : was primarily use to address the imbalance
[15] Vehicle_category : Had over 10 categories
[16] Common sense was used where icy roads were considered as hazardous.

```
class road_surface(CategoryBaseEnum):
    '''Surface condition'''
    NOT_PROVIDED = -1
    NORMAL = 1
    WET = 2
    PUDDLES = 3
    FLOODED = 4
    SNOWY = 5
    MUDDY = 6
    ICY = 7
    GREASE = 8
    OTHER = 9
```

```
109
110 class road_surface(CategoryBaseEnum):
111     '''Surface condition'''
112     UNKNOWN = 0
113     NORMAL = 1
114     MODERATE = 2
115     SEVERE = 3
116
117 class speed_limit(CategoryBaseEnum):
118     '''Maximum speed permitted at the
119
120 class user_category(CategoryBaseEnum):
121     '''User category'''
```

(Figure 6 : Original(L) vs recategorized variable *road_surface*)

More visualizations can be seen either (*EDA-France-Road-Accidents.ipynb or at the end)*.

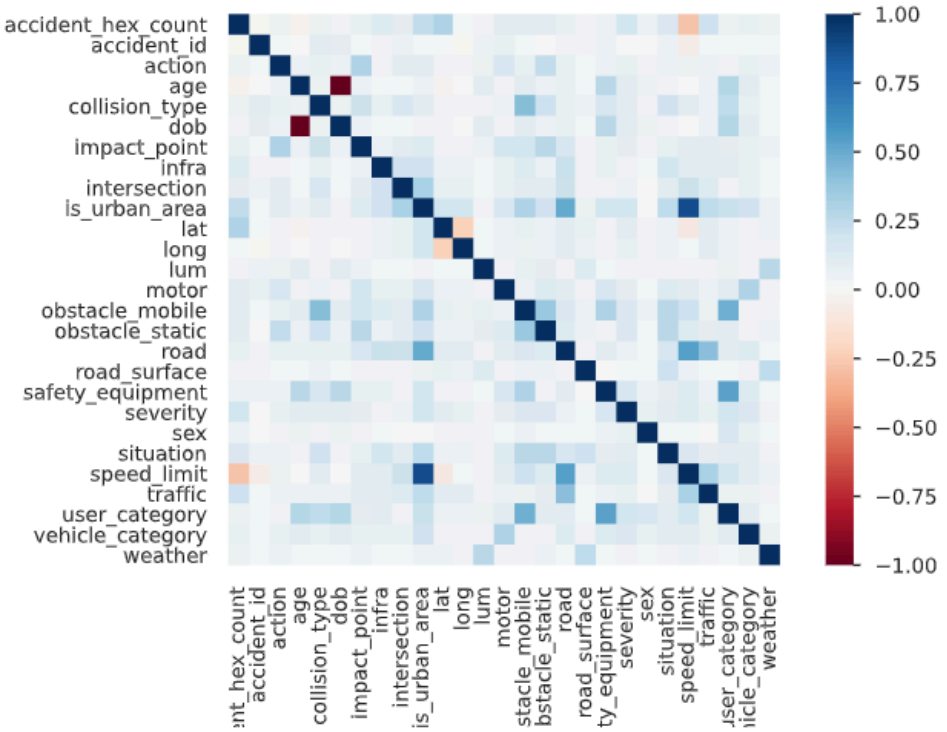


(Figure 5 : Luminosity Bar chart)

Alot of the categorical variables were encoded into simpler manageable categories (see Figure 3) , during this encoding-decoding which was done via our

Enum class we were able to simplify the variables and bring the category count down as well as add an element of hierarchy to the variables impact on the accident fatality.

Once the encoding was done, a simple heatmap correlation (see Figure 7) as part of the EDA was done and it as expected didn't produce any strong connections between the desired target. We noted a few light connections in the form of *lat,long*[17] which correspond to the GPS location of the accident.

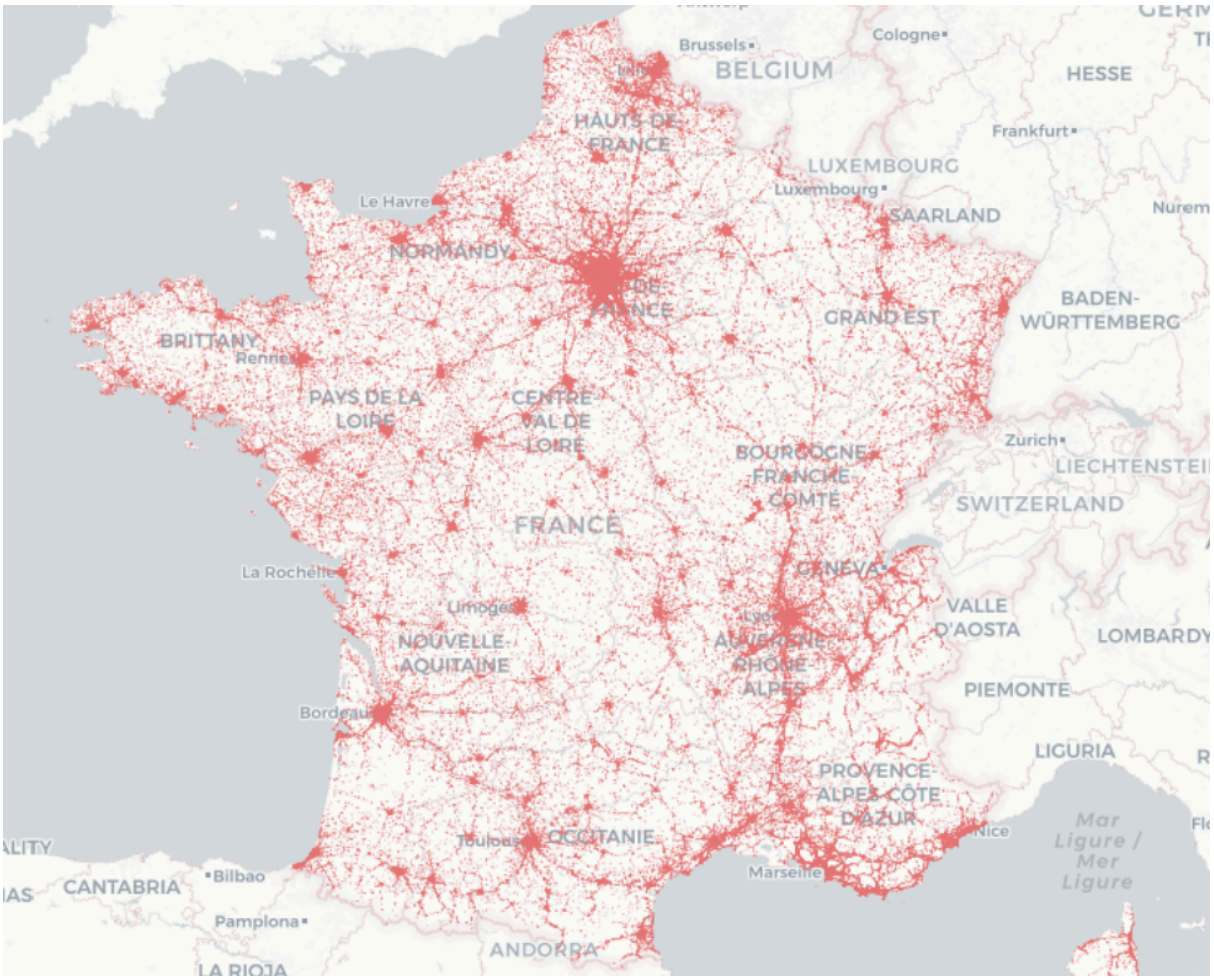


(Figure 7 : Heatmap of the variables)

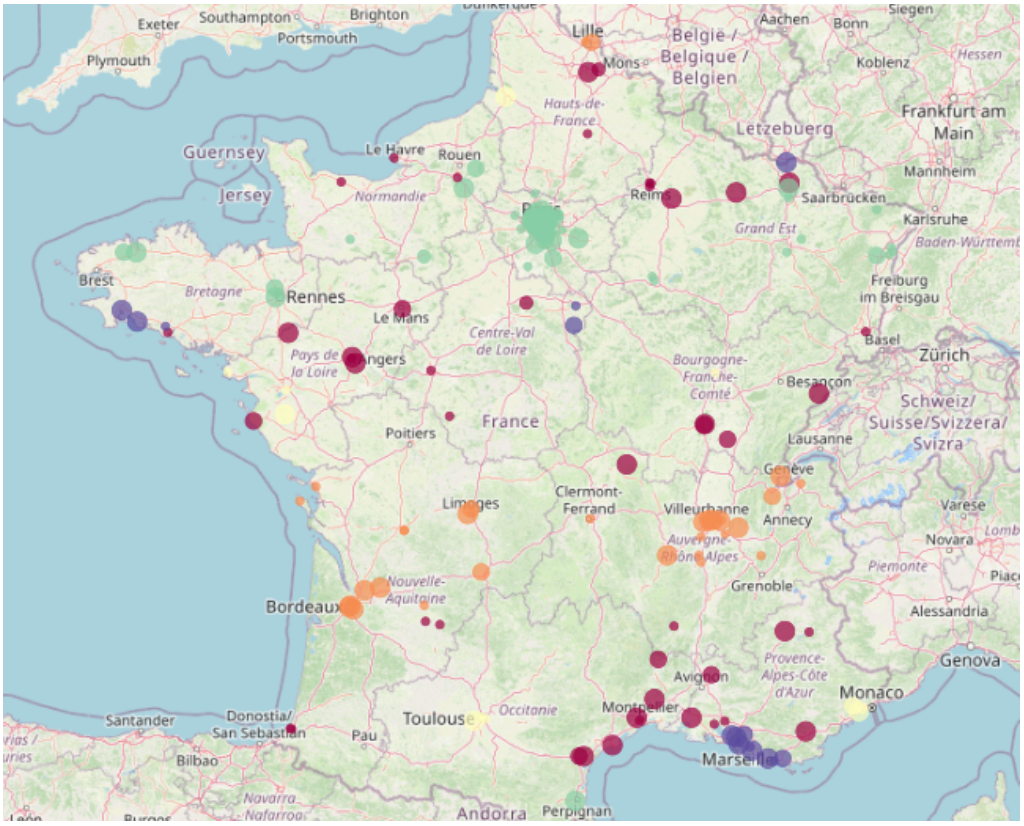
The GPS coordinates prove to be a vital feature and its visualization allows us to see the development and clusters of road accidents in France(see Figure 8).
Our first goal was to naively cluster regions in France according to their accidents and thus create a cluster map of France (see Figure 9) . We used simple and a bit advanced geo-spatial clustering techniques in the form of *KNN & Hierarchical Density Based Spatial Clustering* (HDBSCAN). These naive methods generally provided the main clusters around the metropolitan area of Paris and other large cities but failed to correctly identify outliers in the coordinate data.

[17] I would actually consider based on the plot values to be negligible in correlation and the heatmap almost a naive explanation in this case.

Road Accidents in France



(Figure 8 : Scatter Map of Accidents in France)

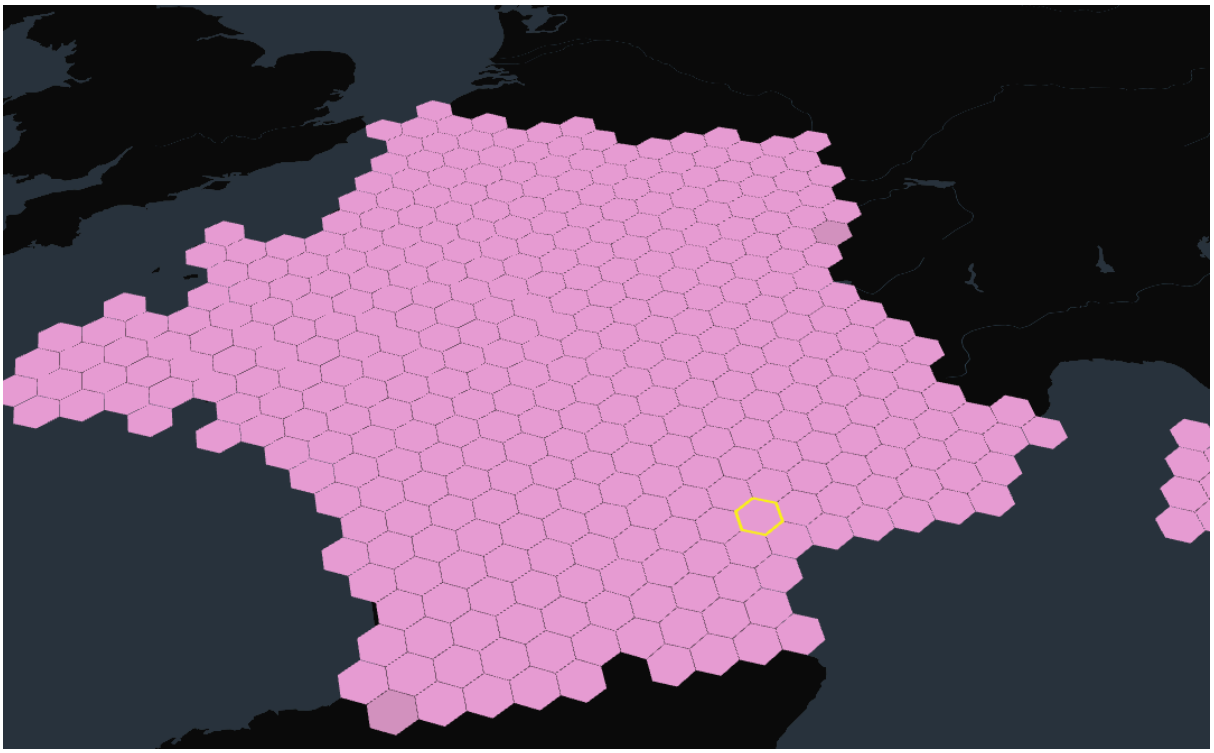


(Figure 9 : Naive Clustering of French Road Accidents)

A better alternative was to use an already available spatial clustering system which would provide much stable and useful results in the form of *H3* (Uber). It creates a hierarchical spatial index that would hash out coordinates[18] into hexagon clusters at a desired resolution which dictates the size of the clustered region. It was quite easy to implement (see below) and thus each Point was hashed into a *h3 id*

```
data = data[mask]
H3_RESOLUTION = 4 # Higher res may cause too many hexsb and that causes kepler gl to crsah
data['h3'] = data.apply(lambda row: h3.latlng_to_cell(row['lat'], row['long'], H3_RESOLUTION), axis=1)
data['centroid'] = data['h3'].apply(lambda x: h3.cell_to_latlng(x))
```

We were thus able to index mainland France and its nearby territory into manageable hexagon areas[19] (see Figure 10) where each hexagon represents an area in the spatial index of France.



(Figure 10 : H3 visualization)

This indexing is computationally expensive hence a lower resolution of 4 was used , however a higher resolution (which means more h3 hexagons) would make sense for inter metropolitan area segmentation especially for Paris.

The *h3*[20] index also allowed us to gather up the accident counts per hexagon which allowed us to keep a tally on h3 hexagons which are more risky (Paris and other metropolitan regions are no surprise given the population density).
The final notable change was to convert the *csv* files to *parquet*[21] for better compression and efficient data storage(from ~200 MB to ~10MB)

[18] Will interchange with Points(Latitude, Longitude)
[19] The H3_RESOLUTION dictates how big the area of the hexagon is
[20] H3 , h3 , hex_index , h3_index , h3 hash , h3 string represent the same thing
[21] https://parquet.apache.org/

Road Accidents in France

Report 2: modeling report

Once the data was passed through the EDA process it made sense to begin naive methods of machine learning to check the validity of one's experiment. We also performed scientific literature research of the recent and state of the art developments in MDL in road traffic analysis and predictions (Behboudi).

(Behboudi) summarizes the main tracks in the space of road traffic analysis as follows

- Accident Risk Prediction (How likely an accident will occur)
- Accident Severity Prediction
- Accident Forecasting / Accident Duration Prediction

Our goals went along the track of predicting the severity of the accident and the authors provided a meta summary of advancements in this area categorized by Machine and Deep Learning. Machine Learning models included *Random Forest Classifier (RF)* and *Gradient Boosting* showed promising results and became the first port of call to try out our experiment. The deep learning models however over time provided better and transferable results and our limitations of hardware pushed us to try a simple *RF* model.

At the same time , we also realized that even though classification was a requirement as dictated in the objectives of the project , any classifier we produce would not end up in production as the idea to classify the severity based on a few highly imbalanced dataset would prove futile and quite naive.

I preferred the idea of Forecasting as it made sense to analyse the yearly patterns of the accidents to predict trends and provide a reliable Forecast. This Time Series analysis could actually provide spatio-temporal insights into traffic flow control over the period of the year and coupled with weather and other variables give live traffic flow control allowing for decrease of fatal accidents. This idea distilled much easier was to see if we could find patterns in the frequency of accidents and provide a model than can predict these patterns by providing a future forecast.

Thus our project provides two core models , a *Future Forecaster* and a *Classifier* which predicts the severity of the accident

Road Accidents in France

Severity Classifier

Classification of the problem:

The severity classifier describes a model capable of predicting the severity of the road accident when provided with data relating to *Point*[22], *vehicle_category*, *weather*, *lum* etc.
We first decided to classify the target as either being a *Major injury* or a *Minor injury.*[23] which would address a bit of the imbalance. Even with this imbalance we decided to use a sampling method ("SMOTE").
After resampling the data we split the data into *train & test* and then used a classification report[24] to gather up metrics of the model.

Model choice and optimization

I experimented with two main models as suggested by the literature overview in the form of a *Random Forest Classifier & Gradient Boosting Classifier*.
A naive understanding of simple classifiers would dictate that in most simple cases a *DTC(Decision Tree Classifier) or RF* would be sufficient to provide adequate results when limited by compute power and hardware resources.

As I was hindered by computation power it made sense to use a hyperparameter optimization technique in the form of *RandomizedSearchCV*[25] build a simple model as seen in the code snippet below. We can also see the params used for creating the different model.

```
tps://stackoverflow.com/questions/39662398/scikit-learn-output-metrics-classification-report-into-csv-tab
rando_classifier(X_train,X_test,y_train,y_test):
params = {
    'n_estimators': [50,100,200,300],
    'max_depth': [1,2,5,10,None]
}
random_search = RandomizedSearchCV(RandomForestClassifier(random_state=random_state),cv=3,n_jobs=-1,pa
random_search.fit(X_train,y_train)
print(f"The best parameters: {random_search.best_params_}")
```

Ideally a brute force method(GridSearch) could provide a better model but we will proceed with the RandomSearchCV which provided decent enough results for a prototype project but would not pass my standard to deploy into production.

An LSTM[26] DLModel as discussed by (Behboudi) would have provided the best results if we had the necessary resources. The advancements in scoring would not however provide cost effect results for the earned accuracy gain. Any other models(CatBoost, XGBoost etc) are all limited by computational power and the negligible resources provided by DS.

---

[22] Point = (Latitude, Longitude)
[23] Classifying only fatalities would be futile even with sampling methods
[24] Accuracy, Recall , F1 score being among the metrics
[25] A greedier option is the HalvingRandomizedSearchCV
[26] Long Short Term Memory model

RandomForestClassifier
n_estimators : 200 max_depth : 10 criterion : 'gini'

GradientBoostClassifier
n_estimators : 200 max_depth : 10 learning_rate = 0.1

(Figure 11 : RF and GB Classifiers)

Results:

We created as discused , 2 main Classifier models (see Figure 11) and the output of the *RandomizedSearchCV* were as follows

	precision	recall	f1-score	support
0	0.764335278261905	0.742114501329578	0.753061006825939	95143
1	0.750035656798223	0.771783797383428	0.760754327047275	95392
accuracy	0.756968535964521	0.756968535964521	0.756968535964521	0.756968535964521
macro avg	0.757185467530064	0.756949149356503	0.756907666936607	190535
weighted avg	0.757176123824854	0.756968535964521	0.756912693930952	190535

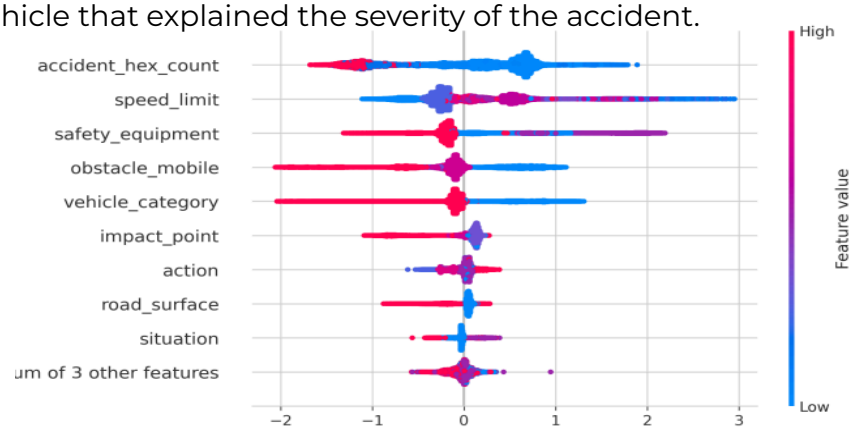
(Figure 12 : Random Forest Classifier Classification Report)

	precision	recall	f1-score	support
0	0.830988297726144	0.826219480150931	0.828597027511331	95143
1	0.827659530113198	0.83239684669574	0.830021428944755	95392
accuracy	0.829312199858294	0.829312199858294	0.829312199858294	0.829312199858294
macro avg	0.829323913919671	0.829308163423335	0.829309228228043	190535
weighted avg	0.8293217388255	0.829312199858294	0.829310158965065	190535

(Figure 13 : Gradient Boost Classifier Classification Report)

The two figures show that the *GradientBoostClassifier(GBC)* performed quite well with an acceptable over *80 percent* score values for both $\{Major: 1, Minor: 0\}$ values[27].

We also provide a *Beeswarm plot* of the SHAP[28] analysis of the *GBC* model(see Figure 14). We note the number of accidents , the type of obstacle hit and the category of vehicle that explained the severity of the accident.



[27] Although the values are quite impressive for a simple model, note that only 190k were used due to sampling

[28] https://shap.readthedocs.io/en/latest/index.html

(Figure 14 : Shap Beeswarm plot of GBC)

TimeSeries Forecasting

Classification of the problem:

Given the limited computation resources and the prototype nature of the classifier , I preferred to use a forecaster to predict the annual cycle of accidents . The time series models would take into account seasonal changes and the effects of *summer, winter, spring, autumn* as well as the effects of weekends and holidays to see the trend of accidents over time.

The forecasting model would take into account these variables to predict the *number of accidents* per annual cycle and would be able to create a future forecast for a user defined period. These could be quite useful in dictating the traffic flow regulation that can be enacted to mitigate the number of accidents during high periods of accidents. All of the data was quite regular except 2020 , which can be explained due to the COVID pandemic which occurred during that period.

The data was then prepared for a time series analysis by selecting features that would add as a regressor(*weather, road_surface, lum)* to the time series model as well as categorizing the accident temporal nature into seasons (See code snippet below, For full code see Forecasting Road Accident.ipynb)

```
# https://neuralprophet.com/how-to-guides/application-examples/energy_tool.html
df["summer"] = 0
df.loc[df["ds"].dt.month.isin([6, 7, 8]), "summer"] = 1
df["winter"] = 0
df.loc[df["ds"].dt.month.isin([12, 1, 2]), "winter"] = 1
df["fall"] = 0
df.loc[df["ds"].dt.month.isin([9, 10, 11]), "fall"] = 1
df["spring"] = 0
df.loc[df["ds"].dt.month.isin([3, 4, 5]), "spring"] = 1

# https://stackoverflow.com/questions/71339576/select-data-based-on-weekday-and-weekend-pandas
df['weekend'] = 0
df.loc[df['ds'].dt.dayofweek.isin([5, 6]), "weekend"] = 1
```

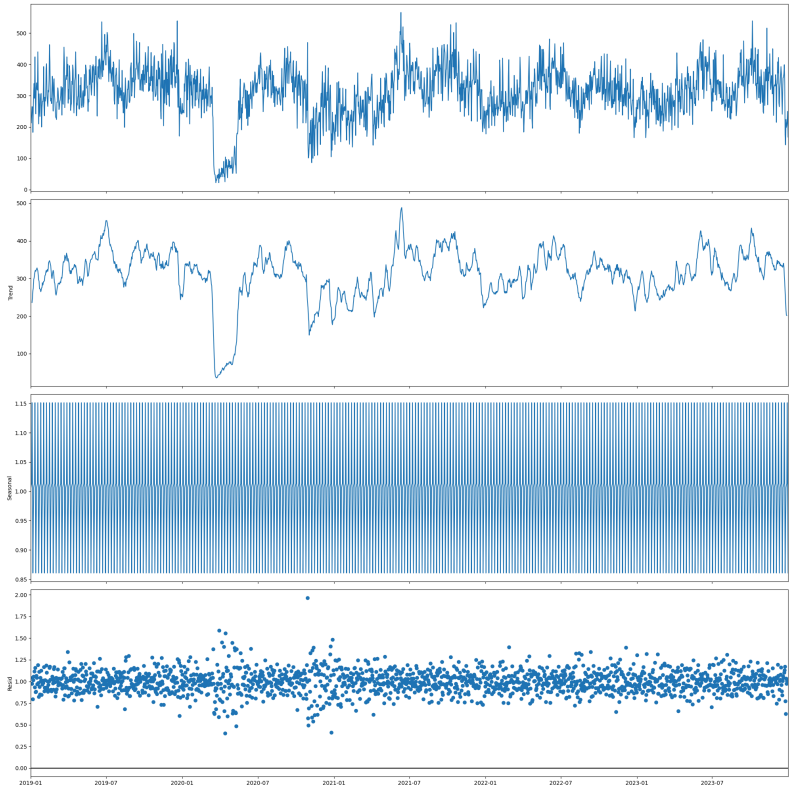
Model choice and optimization

I experimented with two main models as suggested by the literature overview in the form of *NeuralProphet* and *AutoArima with seasonality.*

The ARIMA[29] (“pmdarima”) model with seasonality was the first port to call to tackle this problem. Before we began with the model a seasonal decomposition

[29] Auto regressive integrated moving average (ARIMA)

led us to decide on what type of model we required (see Figure 15). An *ADF*[30] test for stationarity provided that we didn't have to do any more modification to make it stationary



(Figure 15 : Mul Seasonal Decomposition)

The ARIMA model can be summarized by the following code snippet below where the necessary parameters can be seen (For more info see monolith.py)

```
mape_scorer = make_scorer(mean_absolute_percentage_error, greater_is_better=False)
model_arima = auto_arima(train[['y']], start_p=1, start_q=1, test='adf',
                         seasonal=True, m=12, seasonal_test='ocsb',
                         d=None, D=1,
                         trace=True,
                         error_action='ignore',
                         suppress_warnings=True,
                         stepwise=True,
                         maxiter=1,##change higher for real
                         start_P=0, n_jobs=-1, random_state=42, scoring=mape_scorer)
```

The second and significantly better choice was the *NP*(“NeuralProphet”) model which not only had easier features for adding seasonality and regressors but one could also add holidays to the model for consideration.

The model is constructed as below, where the parameters could be tweaked as per the score achieved by the forecaster.

[30] Augmented Dickey Fuller Test (ADF)
https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.ADFTest.html?highlight=adf

It was also noted that *NP* offered a temporal split which meant that the sample space could be split over time with the last six months of 2023 as the validation set and the samples from 2019-mid 2023 forming the basis of the test.

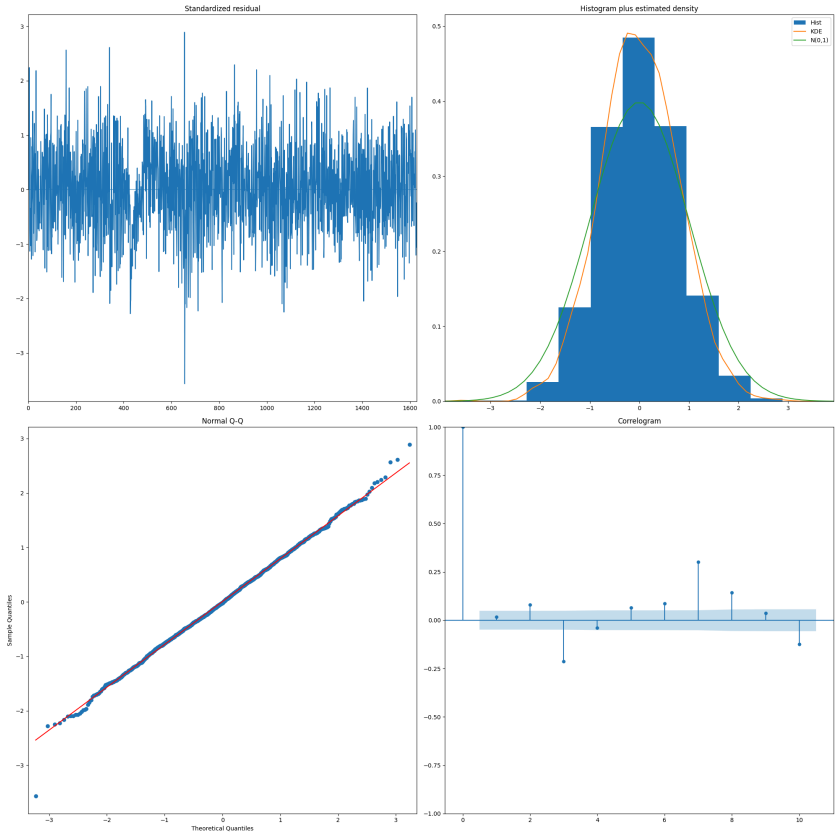
```
%%time
def step_neural_prophet(df):
    df = df[['ds','y','summer','winter','fall','spr
    set_log_level("ERROR")

    m = NeuralProphet(
        n_changepoints=10,
        seasonality_mode='multiplicative',
        yearly_seasonality=True,
        weekly_seasonality=True,
        daily_seasonality=False,
        n_lags=25
    )
    m.set_plotting_backend("matplotlib")

    m = m.add_country_holidays(country_name="FR")
```

Results:

The ARIMA Model provides the diagnostic plot (see Figure 16 & 17) and a summary of the model statistics.



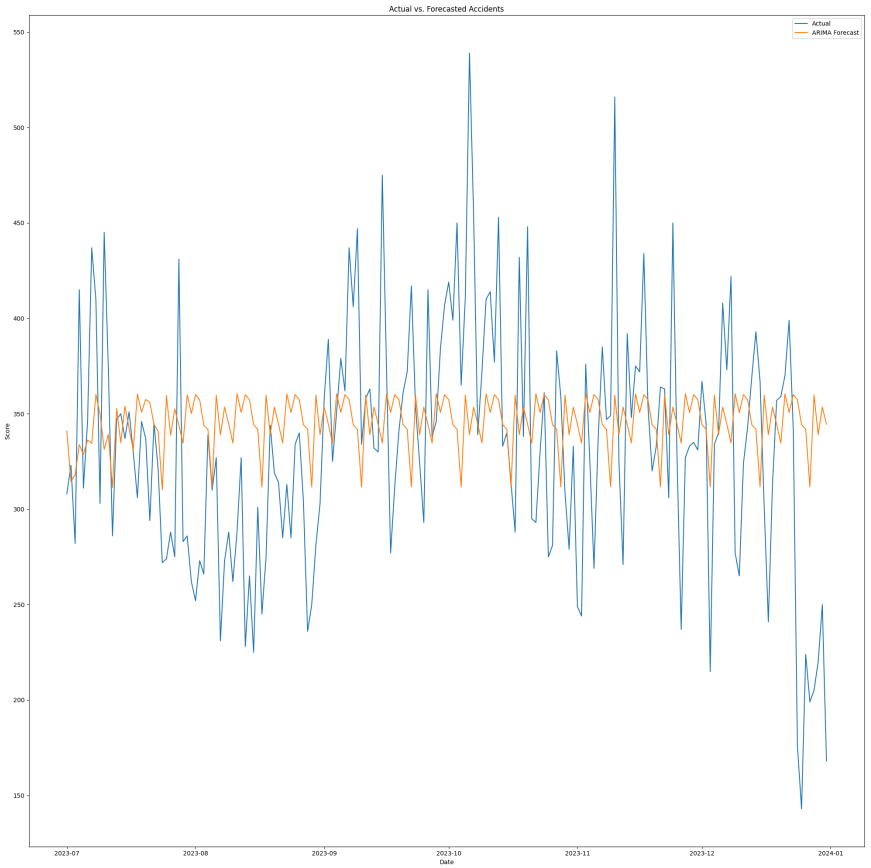
(Figure 16 : Auto Arima Diagnostic Plot)

```
MAPE for time series score: 0.16604221972702474

Model Summary:
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                 1642
Model:             SARIMAX(1, 0, 0)x(0, 1, [1], 12)   Log Likelihood               -9114.090
Date:                              Wed, 05 Feb 2025   AIC                          18234.181
Time:                                      15:00:45   BIC                          18250.370
Sample:                                           0   HQIC                         18240.187
                                                - 1642
Covariance Type:                                opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7958      0.031     25.283      0.000       0.734       0.858
ma.S.L12      -0.7658      0.035    -21.720      0.000      -0.835      -0.697
sigma2      6182.5513    326.301     18.947      0.000    5543.013    6822.090
===================================================================================
Ljung-Box (L1) (Q):                  73.92   Jarque-Bera (JB):                 5.80
Prob(Q):                              0.00   Prob(JB):                         0.05
Heteroskedasticity (H):               0.91   Skew:                             0.04
Prob(H) (two-sided):                  0.29   Kurtosis:                         3.28
===================================================================================
```

(Figure 17 : MAPE & Summary of the model)

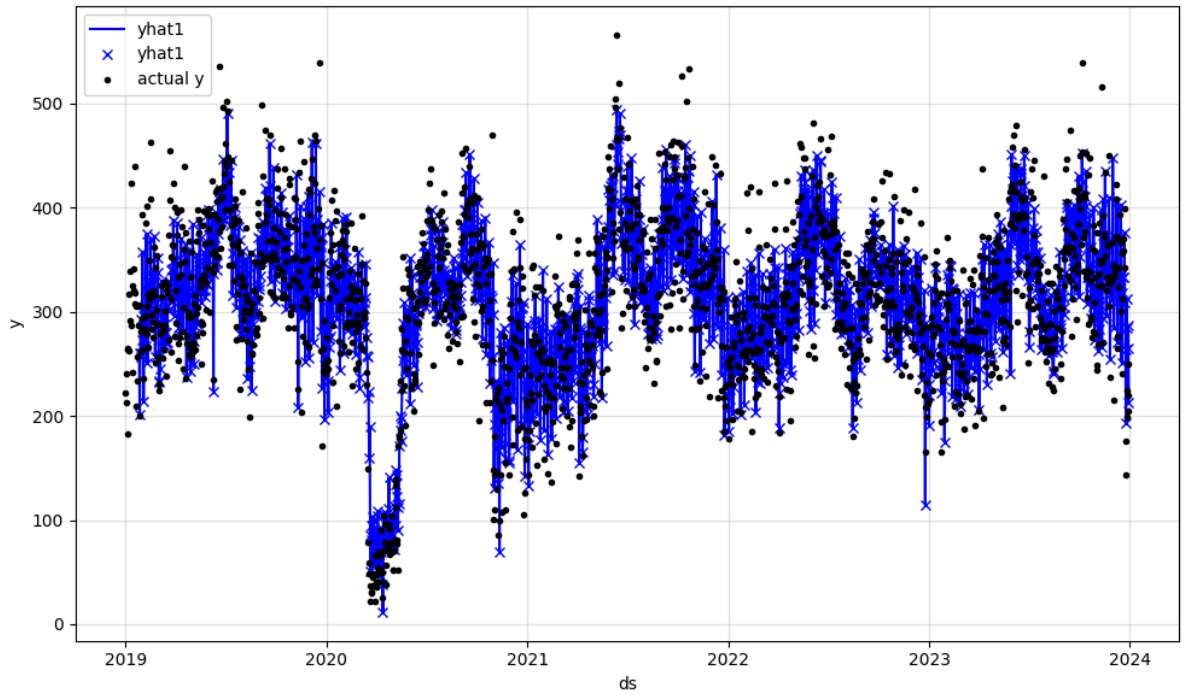
As seen in the figures the model performed quite well and gave an acceptable *MAPE*[31] score of under 20 percent. We can also use the model to create a future forecast and measure its efficacy with the validation set(See Figure 18)



(Figure 18 Actual vs Forecasted Accident Predictions)

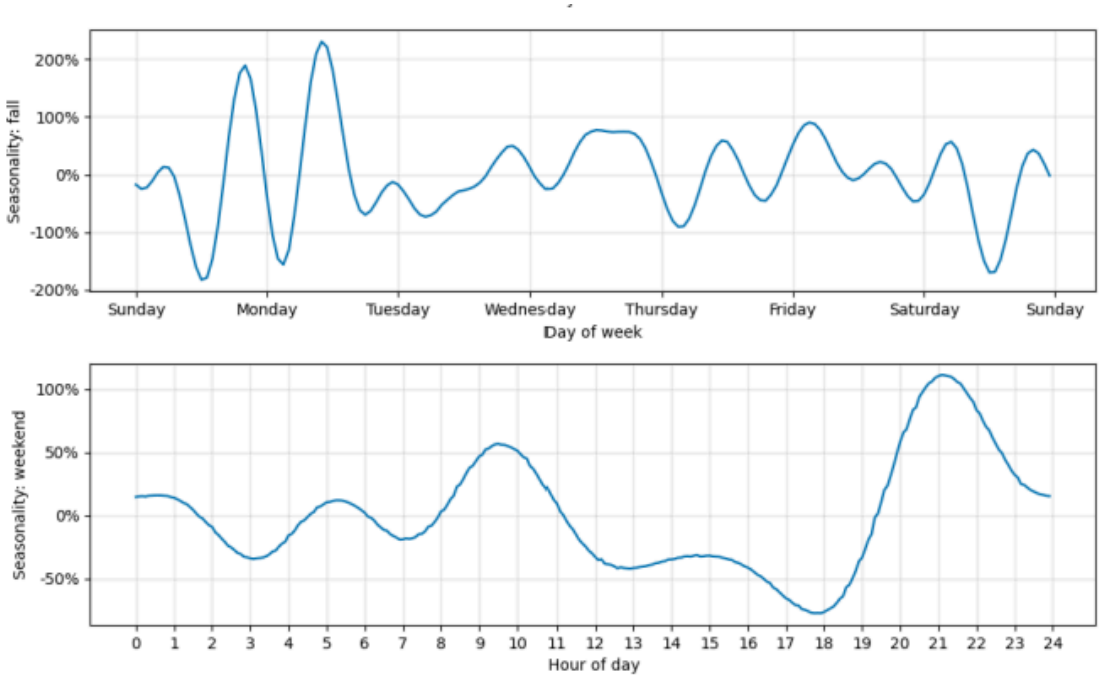
[31] Mean Absolute Percentage Error (MAPE) https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_percentage_error.html#sklearn.metrics.mean_absolute_percentage_error

The neural prophet on the other hand had a better forecasting fit as seen as from below (see Figure 19). Although there are some residuals the fit was much tighter.

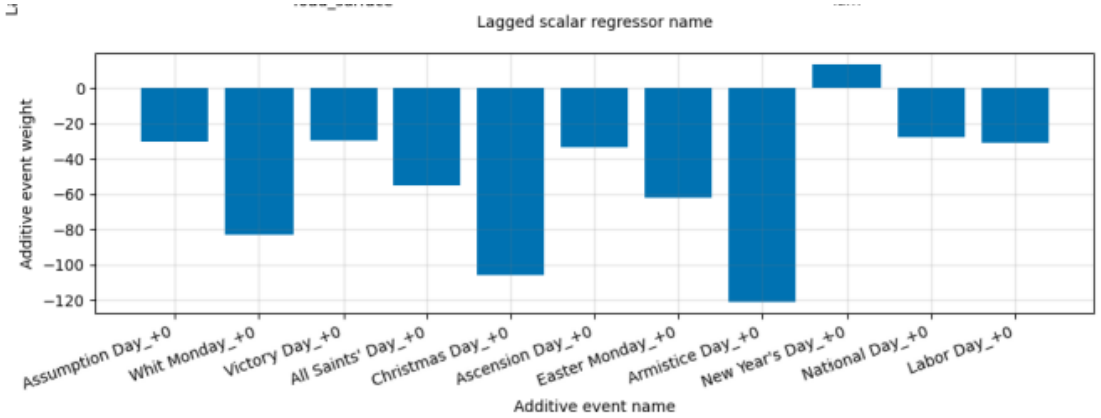


(Figure 19 Neural Prophet Predicted vs Actual Road Accidents)

Since the neural prophet had the added advantage of adding holidays and weekends we can see how the neural prophet used these components to influence the model (see Figure 20 & 21)

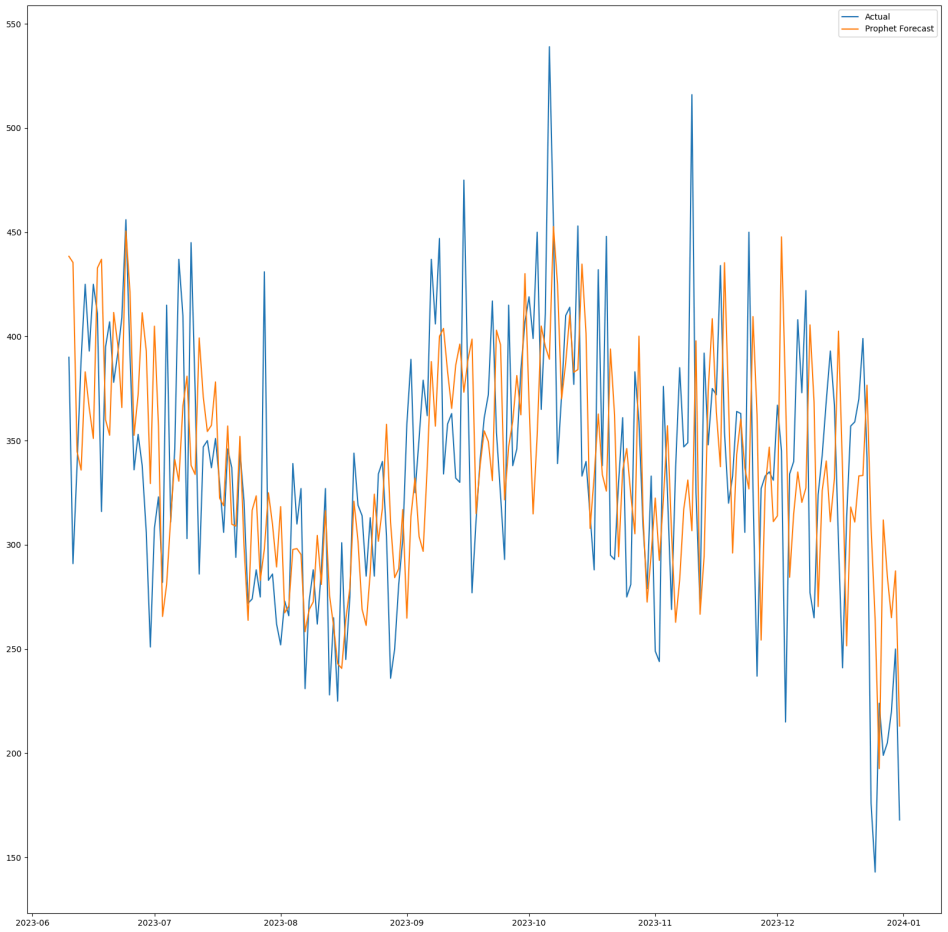


(Figure 20 : Influence of weekend in the season fall as well as the weekend)



(Figure 21 : Holidays in France as an additive event)

Neural Prophet produced a similar *MAPE* score of below 20 percent which is quite acceptable for a good forecasting model. See the overlay of Validation vs Forecasted results (Figure 23)

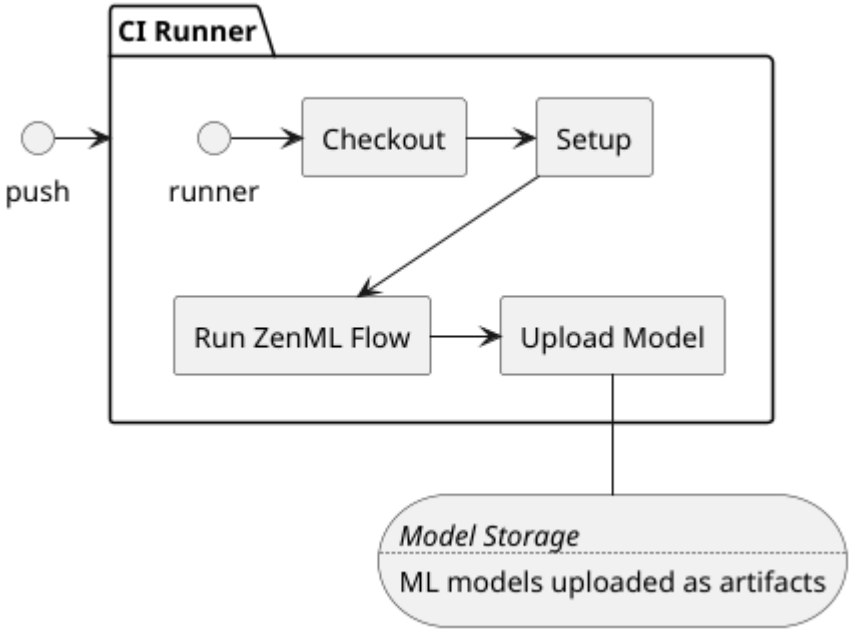


(Figure 22 . Future Forecast vs Validation Set)

Final report :

The experiment both with the severity classifier and the forecaster provided good results for a toy project but in the real world they would be discarded as there is no optimization, no parameter tuning or even allowing the model to run over multiple iterations proved quite tedious as the computational cost was quite high. Given that the DS course costs 12k Euros it made sense to ask for resources & help however none were provided. [32]
Luckily ingenuity and grit are quite natural to me and I decided to set up an automatic Machine Learning system [33]. The idea was to create a CI Pipeline wherein the machine learning model would run and deploy the model as an artifact of the pipeline for later use. This is how one would do even a simple ML project in a production environment. CI/CD allows one to focus on the task of just creating code and not worrying about running the model locally. This would significantly free up my hardware and computational limitations .

So with this in mind and the desire to run longer training models I built a CI/CD Pipeline which allows me to code and not worry about the local training of the model and thus optimizing its parameters (see Figure 23)



(Figure 23 : CI/CD Pipeline Architecture)

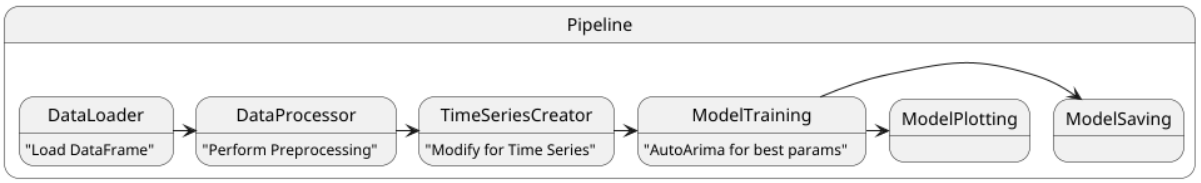
ZenML was chosen as an MLOps[34] orchestrator as it provided both a local and cloud orchestration. The *ZenML* pipeline consists of steps involved in setting up a an automatic Machine Learning Flow for the entire process of loading the data to training the ML Model (see *run-zenml.py)* (See Figure 24 for the steps in the

[32] I asked for an additional space for training and no reply was provided . It is quite understandable that this entire DS course and project was nothing short of incompetence and a minus rating wouldn't even do it justice. The DS mentors and tutors were no help.
[33] A sort of CI/CD for Machine Learning with an automatic training and model deployment system.
[34] MLOps : Machine Learning Ops

pipeline). This pipeline was quite a feat and would allow me to finally focus on creating models and tuning them rather than running the models.

I hit another roadblock in the space provided by DS and even after a request , no reply was given. I am including this in the report as it is important to understand how the DS is extremely poor and it's nothing short of a waste of time.

I would have loved to run a longer training for either the classifier or the time series model but we are sticking with the results we got with the unnecessary hurdles and roadblocks provided.

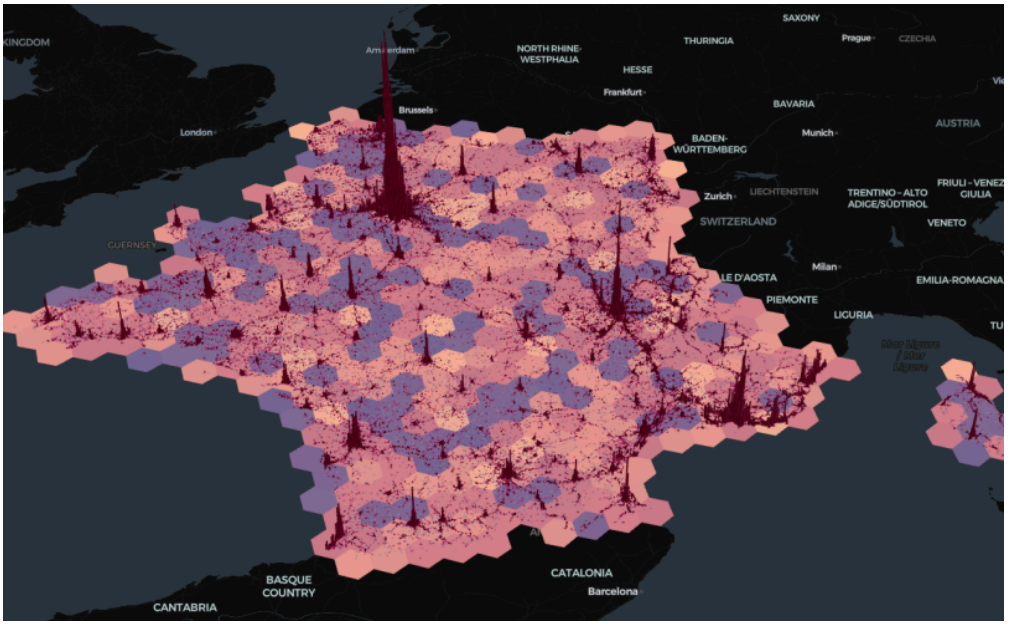


(Figure 24 . ZenML Pipeline)

The goal of the project asked for the creation of *Danger Zones* indicating regions of France which are more risky in terms of traffic flow and its fatalities.
The *Risk Score* was born out of the idea to score each zone according to its risks based on the number of accidents, the type of accidents and other factors contributing to its score where 1 is lower risk and 4 being the highest risk. The *Risk Score* can be simply calculated as the weighted average of factors influencing the likelihood of having a fatality or injury given certain data variables

$$Risk\ Score\ = \alpha\times[Primary\ Factors] + \beta\times[Secondary\ Factors]$$

where the weights can be set with α being a stronger weight , see code for understanding the score. The score then can be used as a visualization tool to see which zones of France are more riskier or dangerous in terms of traffic flow (see Figure 25)



(Figure 25 Danger Zones of France)

Conclusion

The French Road Accident Project was thus concluded by providing three interesting ideas that satisfy the objectives set out by the project.

The severity classifier with RF *& GBC* both being able to have high classification scores. (Behboudi) also shows that these classifiers fit in the simple machine learning models for explaining traffic accidents. They can definitely be improved with hyperparameter optimizations and training but it makes sense in 2025 to use a DL model rather than a simple ML model if one wants to use this as a guide for controlling and negating traffic accidents.

The *NeuralProhet* forecaster was quite interesting and would be more inline with a real world scenario. The addition of more regressors and tuning would significantly improve the *MAPE* score . I think if we run the model and validate it with the 2024 data the *MAPE* score would be comparable and its forecast for 2025 could help in deciding how to change the traffic flow conditions to avoid further fatalities.

The Danger Zones of France as seen in Figure 25 help us thus identify zones in France where the traffic flow needs to be regulated. As expected Paris tops out but due to computational limitations its finer grained resolutions are not visualized. The project has been successfully fulfilled and I am sure the French government or readers of this project report can definitely appropriate some ideas to help in traffic flow decisions and hopefully create policies to decrease the number of traffic fatalities.

References

Behboudi, N. (n.d.). *Recent Advances in Traffic Accident Analysis and Prediction: A Comprehensive Review of Machine Learning Techniques*.

HDBSCAN. (n.d.). *HDBSCAN*. https://hdbscan.readthedocs.io/en/latest/basic_hdbscan.html

Ministry of Interior. (n.d.). *Annual databases of road traffic injuries*. https://www.data.gouv.fr/fr/datasets/53698f4ca3a729239d2036df/

NeuralProphet. (n.d.). https://neuralprophet.com/

pmdarima. (n.d.). https://alkaline-ml.com/pmdarima/index.html

SMOTE. (n.d.).

https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

Uber. (n.d.). *H3*. https://www.uber.com/en-DE/blog/h3/

WHO. (n.d.). Road traffic injuries.

https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries

Code & Kaggle :

The French Road Accident located on Github is the basis of the project where the main branch is *develop* . This is the product of a lot of hard work and prototypes and umpteen amounts of testing code located on Road Accidents.

The main notebooks for the code are located in *notebooks* folder which are hosted on Kaggle.

Handicap & Difficulties in the Project:

1. Although Kaggle datasets were forbidden, this quite often used data set was thrust upon us .This dataset holds nothing unique and was taken from the bottom of a pile of datasources. No forethought was put in by the DS in selecting this Dataset with the criteria of taking unique and field related datasets. This probably stems from the incompetence and a last minute selection of the dataset from the DS team.

2. The data was completely in French with no translations provided and when asked almost 20 days later. Even Though the translation was available to the DS Team as evident from MC SQL and Data Warehousing where the french accident dataset was used again, the translation was not provided even after asking. This also proves point 1 , where even a MC used this toy dataset.

3. The next hurdle was the team composition where being thrust into a team of 4 with a mentor was another pitfall. After creating a project plan with user stories for team based collaboration on Github and Miro board for ideation , it became quite evident that the other team members had no interest in collaboration or taking the abysmal project seriously.[35]

[35] Which is quite understandable , it's a throwaway project , a toy example that I'd definitely not add to my portfolio.

4. There was no direction or no leadership and the mentoring was almost a waste of time . The meetings on Monday were sporadic and were cancelled or changed by the mentor almost at the last minute which showed that the DS team had no intention of putting in effort compared to what I did expect as seen by my report and my project.
5. Nevertheless this gave me a chance to try out my MLOps idea and ideas for using newer and better methods to advance my learning even in the face of lackluster modules provided by the DS platform and team.
6. The biggest slowdown was the translation of the dataset, once it became evident that the other members were almost absent , it freed me up to radically develop my skills and use this toy project as a springboard to actual Machine Learning for a proper DevOps environment. Next slowdown was the computation resources and the lack of help from the DS team to provide additional space for training even though over 12k was paid for the course. [36]
7. It was quite evident that no thought was put into selecting the dataset. Although Kaggle datasets were supposedly forbidden , this quite often used dataset was thrust

8. Any skills used in this project were most likely acquired from free Youtube courses or blogs which provided a better understanding of core concepts.

9. Any hurdles I faced scientifically were more less answered by the literature review (Behboudi) that I did and it made sense to see the futility of a severity classifier , nevertheless as it was required the task was performed.

10. As mentioned , all of the hurdles are mostly from the DS infrastructure and team as the project in itself was a toy experiment and was in terms of code not that complex.

[36] A Google Collab Pro+ costs ~50 Euros a month and in hindsight the compute resources provided along with a GPU would have been an exponentially better purchase.