Accidents In France

Data Mining and Machine Learning

With Visualization

Report

Group 4

16 / 6 / 2019

Content

1 Instruction

From the invention of various types of transportation, the convenience of transportation is accompanied by certain dangers.

Our project focuses on statistics, analysis and prediction of traffic accidents. We hope to get some information from the existing data to help our transportation department make better decisions.

Machine learning is the core technology we use. We mainly use PCA, LDA, t-SNE, Decision Tree, SVR and other algorithms to reduce, cluster and analyze data, using Python as the main language, using Common Python tools such as Pandas, sklearn and matplotlib to help us code.

A variety of rich visualization technologies can visualize abstract figures. On the one hand, we can make the machine learning technology we use more understandable, and on the other hand, the traffic supervision department can better understand the meaning behind the data. We used tools such as Echarts and the Google Maps API to code a series of interactive visualizations based on HTML, CSS, and JavaScript.

We also designed and produced a website that simulates the actual usage scenarios of the traffic department and implements a complete interactive process, including exciting features such as existing data analysis, real-time predictive analysis of data uploads.

2 Timeline & Task Repartition

Name	Tasks
Yang Li **刘洋**	Machine Learning (Clustering,Dimensionality reduction & Predicting) Provide data interface
Zhaohui Li **李朝晖**	Back-end（Data Conversion, 1st Round） Front-end（GUI） Deployment（Cloud Server）
Shuhao He **何书豪**	Document & Slides (Including PRD,prototype) Data Conversion (2nd Round) & Visualization UI/ UX Beautification & Domain Name

Tasks	Duration	Note
- Choose a dataset - Draw UML diagrams	1 week	
- Confirm the technology stack - Configure the environment	3 days	- flask: to build back end. - bootstrap,jquery,html,css: to build the front end. - nginx,gunicorn: to deploy the project. -Tencent Cloud Domain Name Service and DNS
- Learn related knowledge	4 days	- Sklearn：to build Machine Learning models - Pandas：to process the dataset - Google Maps: to show the accidents on maps - Baidu Echarts and Python matplotlib: Visualize the data and prediction - GEOJSON & GEOCODING: data format or method for Maps
- Data processing - Make GUI (The website) - Visualizaiton	2 weeks	Please turn to **Subtask List** below for subtasks
- Docs and PowerPoint Slides	3 days	
Total	**Around 38 days**	

Attached Table: Subtask List

Sections	To do List	Note
Data processing	· Data preprocessing (4-5 Days) · Prediction (3-4 Days) · Analyze data (3-4 Days) · Provide data interface(2-3 Days)	
GUI (The Websites)	· GUI (7-11 Days) · Interaction between front and back end (2-3 Days) ·Translation (2-3 Days) ·Web Server deployment (3-4 Days)	
Visualization	· Basic Maps: 4 Days · Interactive Map: 4 Days · All kinds of charts: 6 Days	A part of Data Processing is included in Visualization

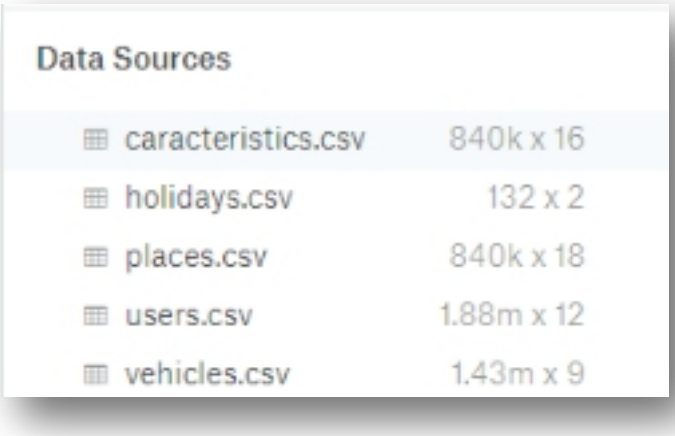
3 Initialization step (Step I)

i. Choose the data

The dataset selected by our group is "accidents-in-france-from-2005-to-2016"

Url:https://www.kaggle.com/ahmedlahlou/accidents-in-france-from-2005-to-2016

In the dataset, there are 5 csv files, which are caracteristics.csv, places.csv, users.csv, holidays.csv and vehicles.csv.

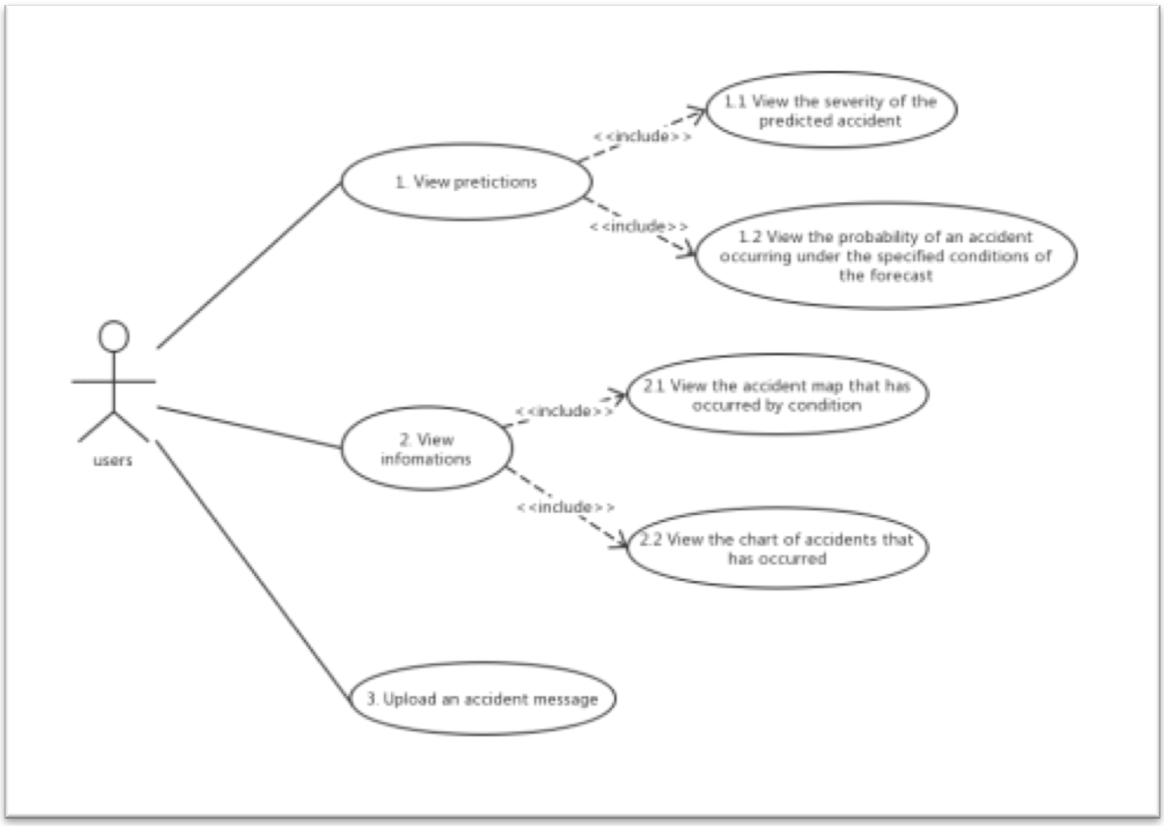


These files detail the properties, accident locations, personnel and vehicles of every traffic accident that occurred in France between 2005 and 2016.

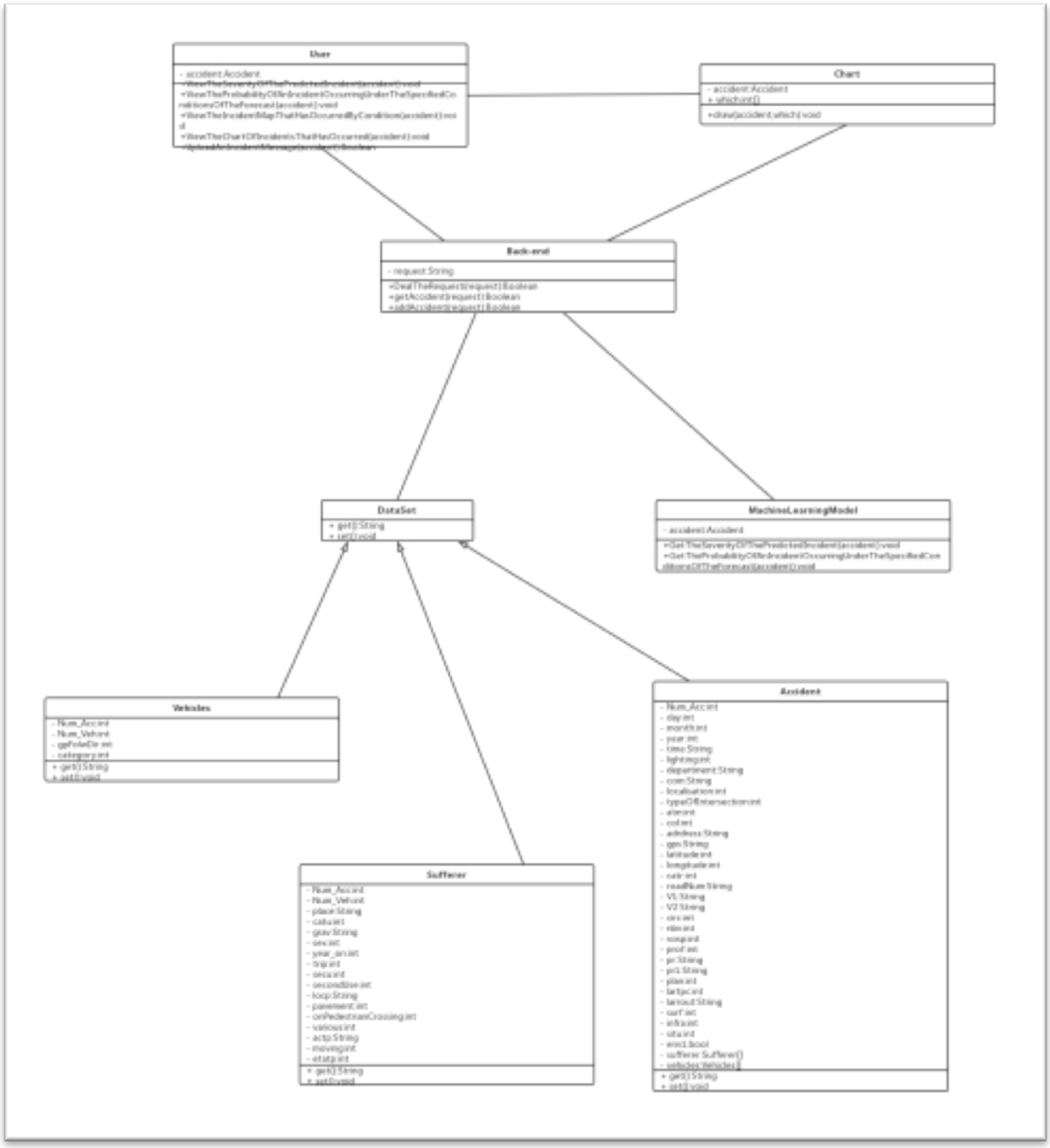
What we want to do is to use machine learning to predict the likelihood of future traffic accidents and the number of deaths in traffic accidents that have occurred by processing the data set. In addition, we will visualize existing data to analyze the impact of different factors on traffic accidents. Our team is ready to create a web-based app based on the above ideas.

ii. Elaborate the Use-case diagram and detailed description of the most important cases

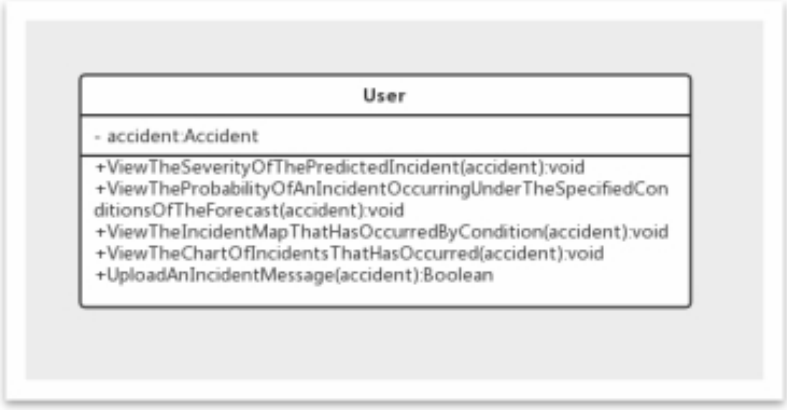
- Use Case Diagram



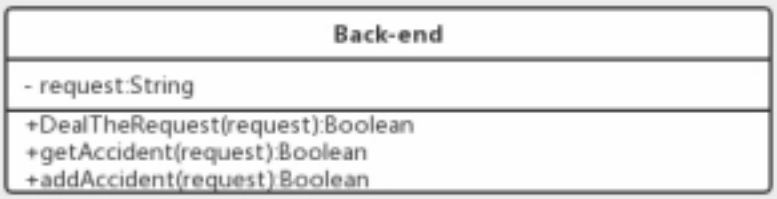
- Class Diagram



- Users class

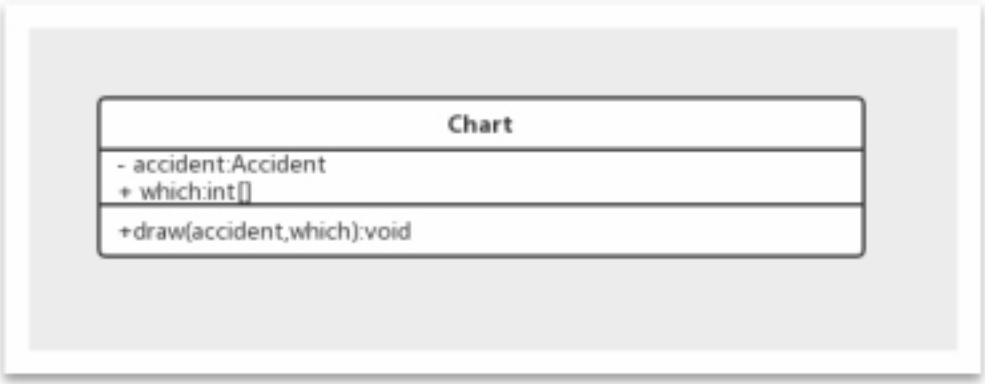


- Back-end class

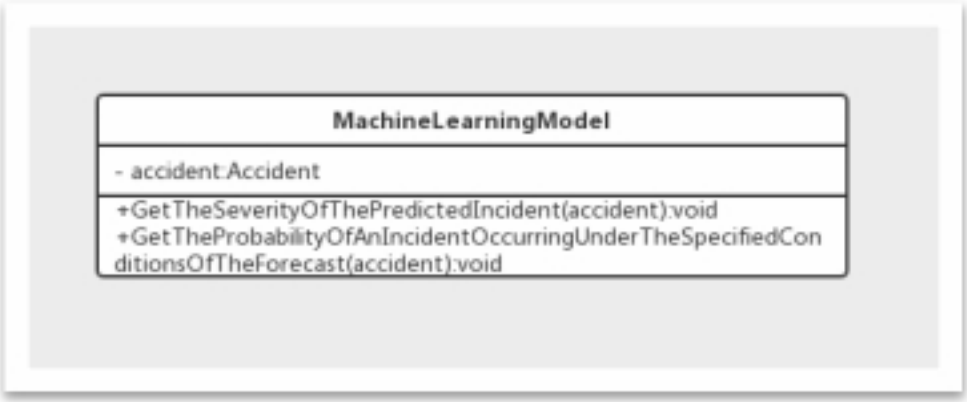


- Accident class

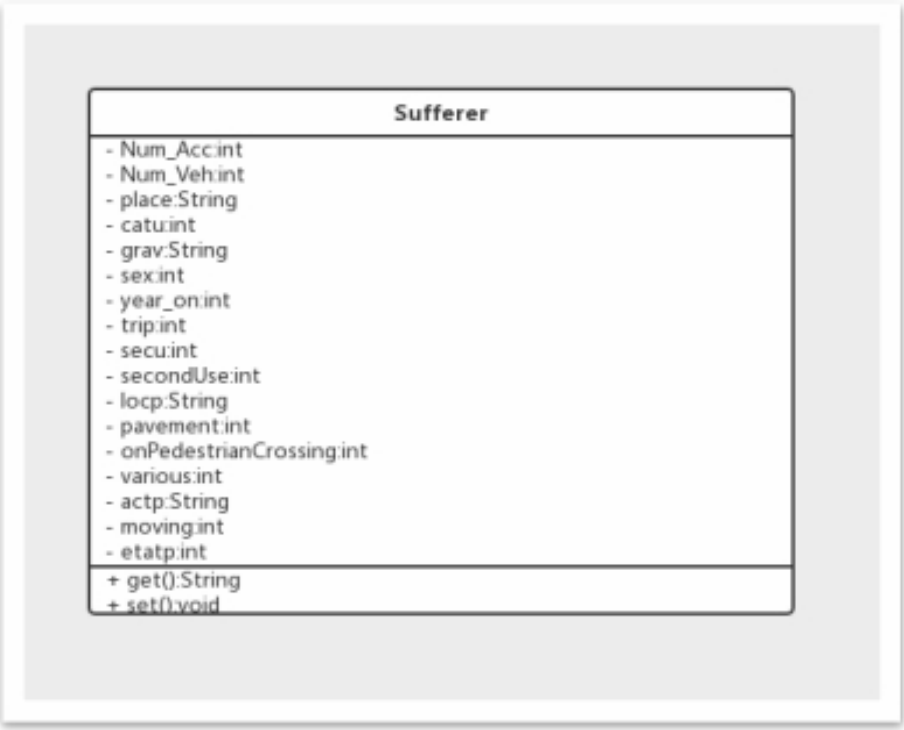
- Chart class



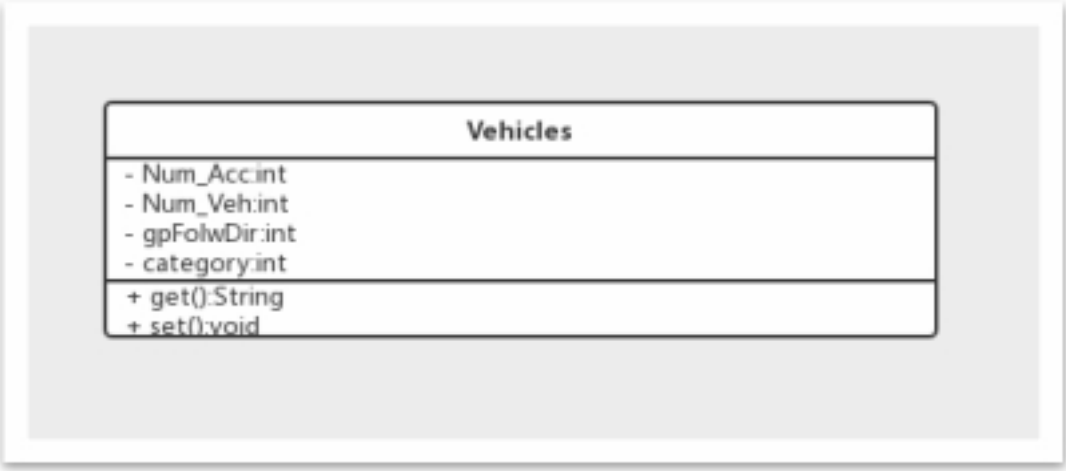
- MachineLearingModel class



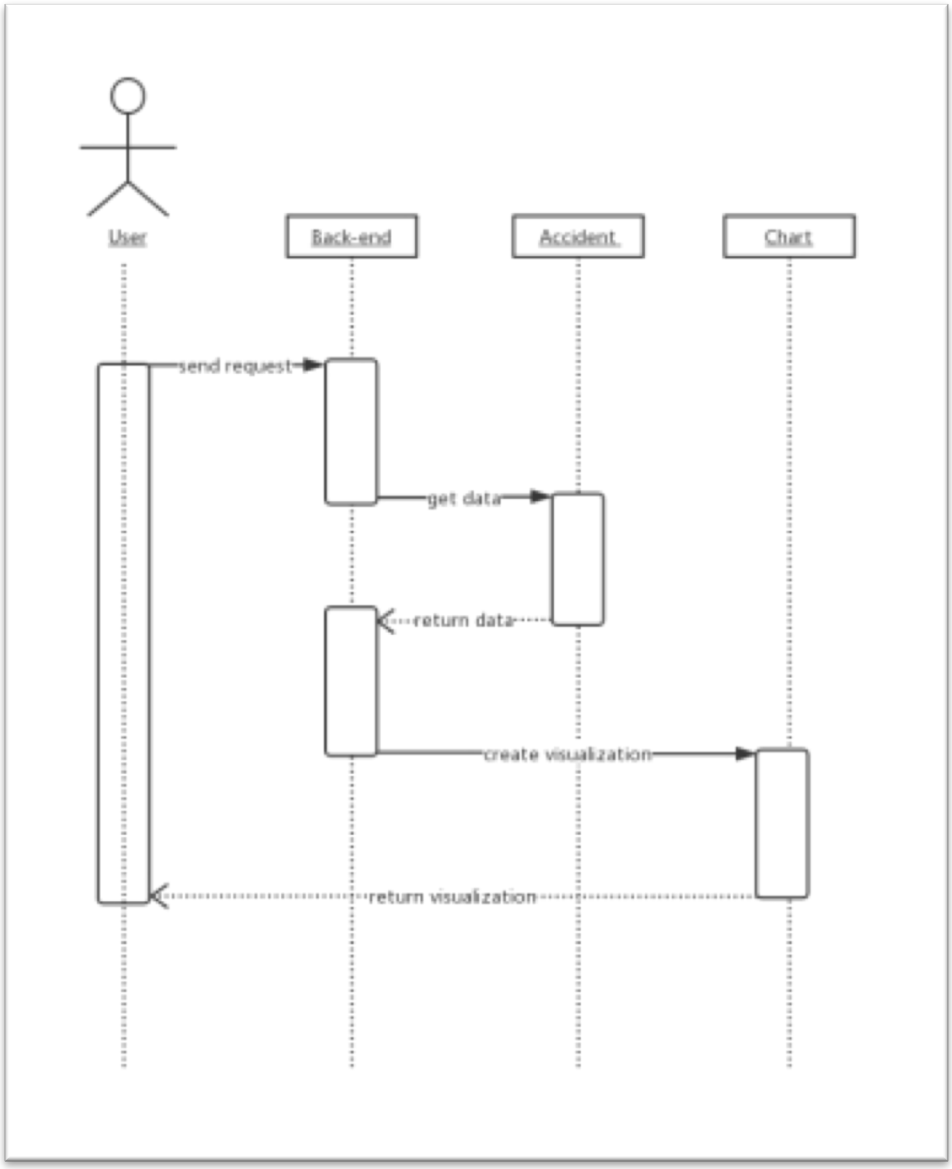
- Sufferer class



- Vehicles class

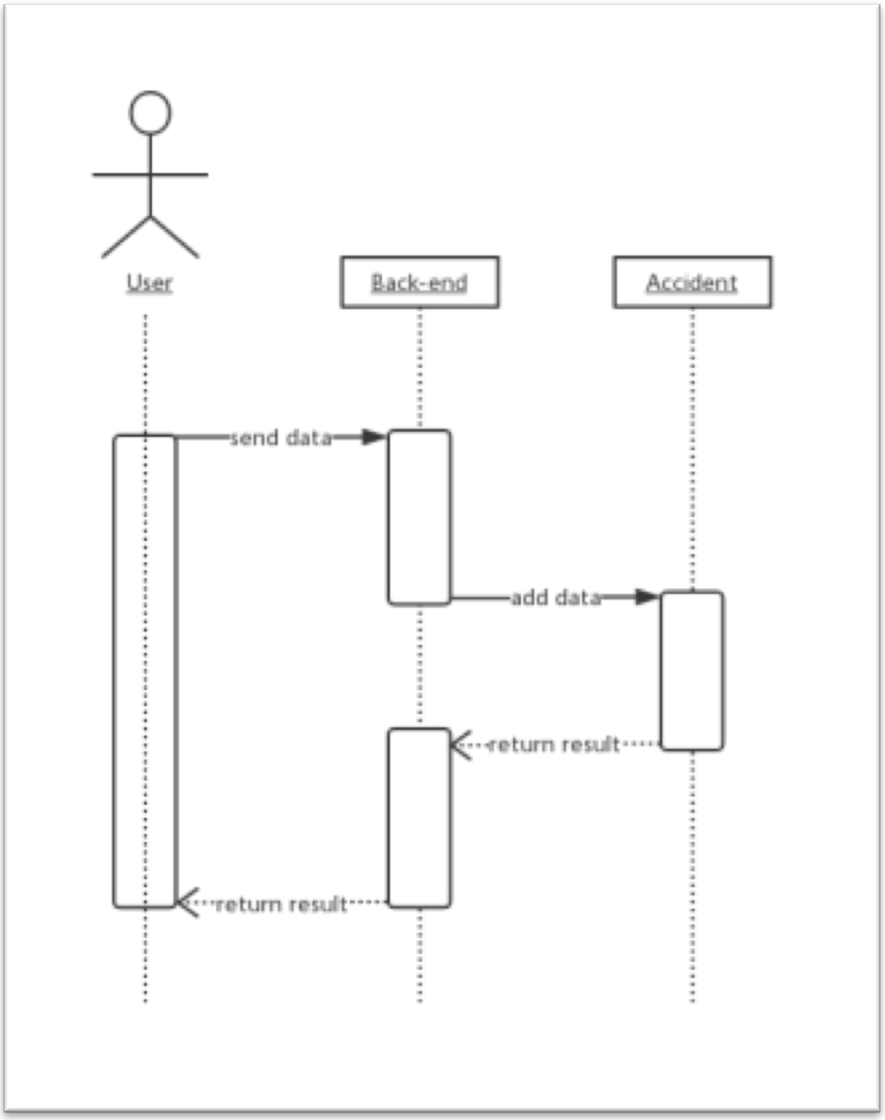


- Sequence Diagram



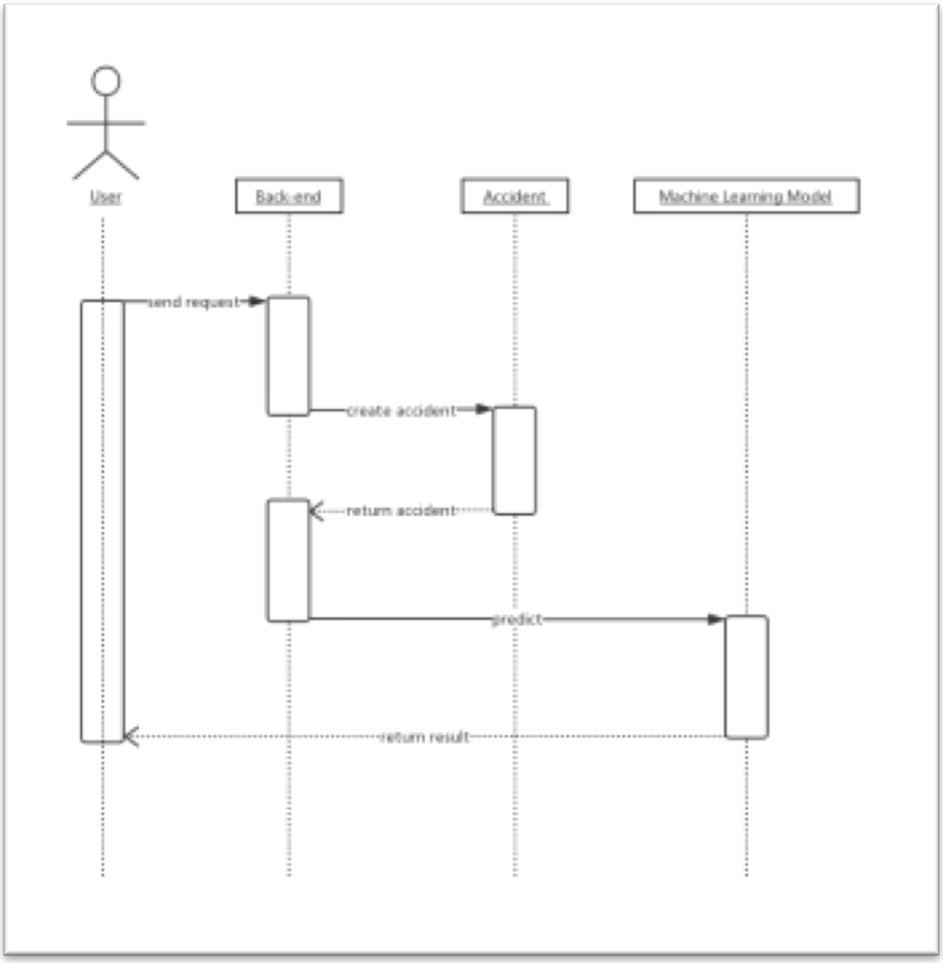
- Visualize

Users can query the specified range of accidents, then get a visual map.



- Upload

Users can upload accidents through our system.



- Predict

Predict the probability of an accident occurring under specified conditions

Predict the severity of an incident.

iii. Define the global architecture of the Project

Our system is based on B/S framwork, so officers can use it accessibly by both smartphone and PC.In order to be international, our system will support 3 languages: French, English and Chinese.

4 Elaboration step (Step II)

i. Detailed architecture of the Project by describing all the functionalities and the employed languages;

Functionalities :

Predict the severity of the accident (under the premise of an accident)

Predict the probability of an accident occurring under specified conditions. (Conditions are specified)

Query the accident occurrence map according to the conditions

Various charts

Punch Card of accident(Describe the number of people in the week, time and accident)

Visualization With Google Map(Show all traffic accidents,heat map

Clustering map based on coordinates

Pie chart(Number of people injured, killed or slightly injured in traffic accidents in the selected month)

Heat map(Show trends in deaths in each year)

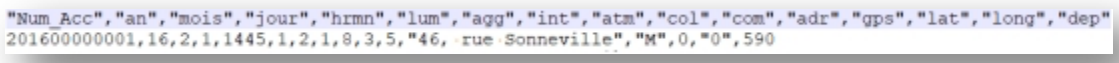
- The employed languages:

Python,HTML,CSS,JS,Nginx

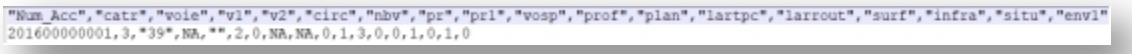
ii. Scraping and collect the data

There are 4 CSV files in our dataset, which are caracteristics.csv, places.csv, users.csv and vehicles.csv.

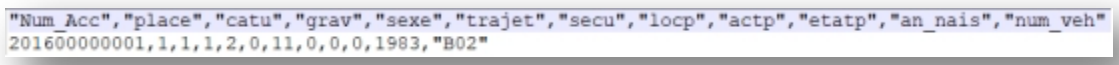
caracteristics.csv:



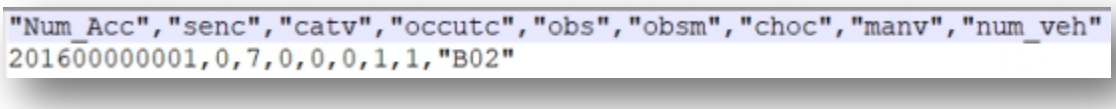
places.csv:



users.csv:



vehicles.csv:



A detailed description of each column is on the website:

https://www.kaggle.com/ahmedlahlou/accidents-in-france-from-2005-to-2016

What we need to do is to predict the probability of an accident based on various environmental factors on the day of the location and to predict the level of casualties based on the accident that has occurred.

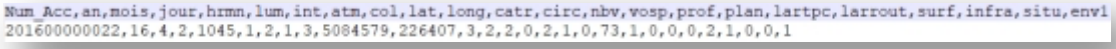
So the most useful of the four files for forecasting are caracteristics.csv and places.csv.

First, merge the two files(caracteristics.csv and places.csv) through Num_Acc.

Second, delete columns that are not related to predicting by pandas.

There are many columns in our dataset that contain all aspects of the incident. But some columns don't help relative to the predictions, such as the address and the commune number.And there are several columns in the dataset that don' t have an introduction to it on the site, such as 'agg'.

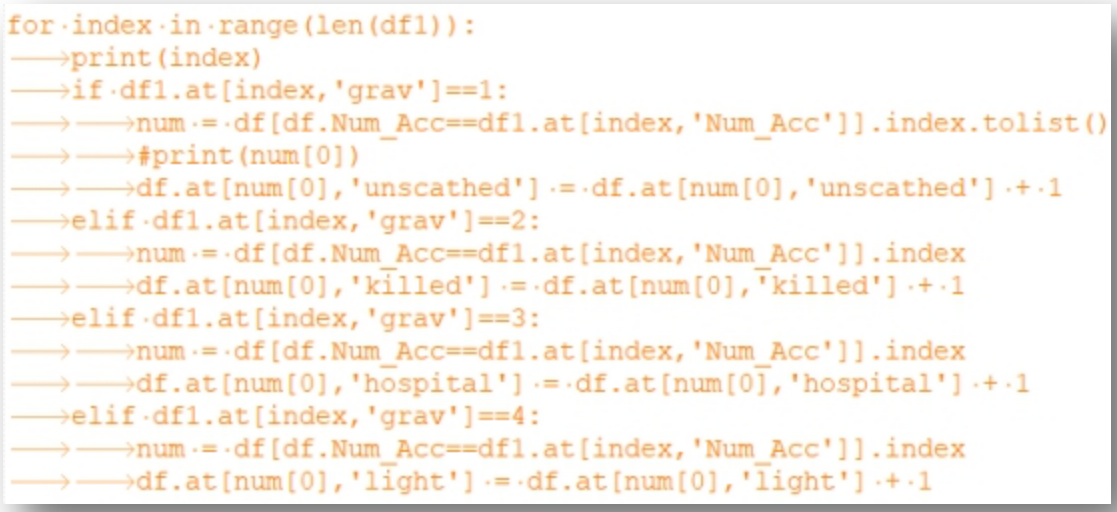
So I deleted these columns using pandas and got a new CSV file(by deal.py):



```
Num_Acc,an,mois,jour,hrmn,lum,int,atm,col,lat,long,catr,circ,nbv,vosp,prof,plan,lartpc,larrout,surf,infra,situ,env1
201600000022,16,4,2,1045,1,2,1,3,5084579,226407,3,2,2,0,2,1,0,73,1,0,0,0,2,1,0,0,1
```

Third, count the number of people with different degrees of injury (unscathed, killed, hospital, light) in the 'users.csv' to the accident (caracteristics.csv').

One attribute in the users.csv file is grav, which marks the injury status of everyone in each accident. An accident usually corresponds to several related people. I need to count the number of people injured in each accident. So I use deal. Py related code to count the number of injured in each accident.



```
for index in range(len(df1)):
    print(index)
    if df1.at[index,'grav']==1:
        num = df[df.Num_Acc==df1.at[index,'Num_Acc']].index.tolist()
        #print(num[0])
        df.at[num[0],'unscathed'] = df.at[num[0],'unscathed'] + 1
    elif df1.at[index,'grav']==2:
        num = df[df.Num_Acc==df1.at[index,'Num_Acc']].index
        df.at[num[0],'killed'] = df.at[num[0],'killed'] + 1
    elif df1.at[index,'grav']==3:
        num = df[df.Num_Acc==df1.at[index,'Num_Acc']].index
        df.at[num[0],'hospital'] = df.at[num[0],'hospital'] + 1
    elif df1.at[index,'grav']==4:
        num = df[df.Num_Acc==df1.at[index,'Num_Acc']].index
        df.at[num[0],'light'] = df.at[num[0],'light'] + 1
```

Then the new caracteristics.csv:

iii. Data cleaning and transformation

First, delete items with invalid data.

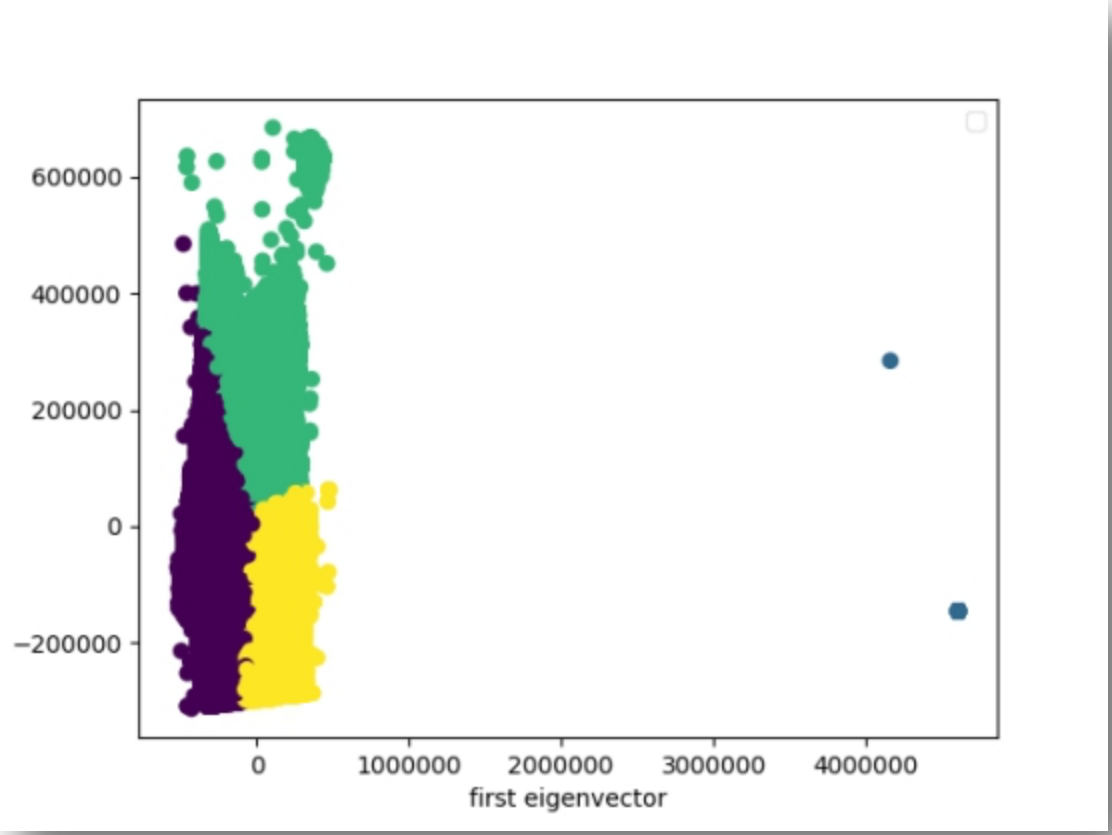
The latitude and longitude column contains a large number of 0 and null, and the other columns also contain a lot of null data. So I need to delete these data by pandas.

Second, Principal component analysis(by PCA.py)

```
num = 2
df = pd.read_csv(url1,encoding="LATIN_1",low_memory=False)#iso8859_15
X = np.array(df[['lum', 'agg', 'int', 'atm',
'col', 'com', 'lat', 'long', 'dep', 'catr', 'circ', 'nbv', 'vosp',
'prof', 'plan', 'lartpc', 'larrout', 'surf', 'infra', 'situ', 'env1']])
pca = decomposition.PCA()
pca.fit(X)
print(pca.explained_variance_)
pca.n_components = num
X_reduced = pca.fit_transform(X)
pc = pd.DataFrame(X_reduced)
pc.to_csv(urlP,encoding="LATIN_1",index=False)
```

```
[3.96744976e+11 3.95238004e+10 6.85048116e+04 2.74009923e+04
 3.10176330e+03 2.35079988e+03 4.04314247e+02 4.25109825e+00
 2.80612061e+00 2.68638521e+00 1.98451265e+00 1.72221013e+00
 1.41687372e+00 1.06412268e+00 8.79259095e-01 6.32082625e-01
 5.01928259e-01 3.79631230e-01 3.67837102e-01 2.54898625e-01
 1.59900040e-01]
```

Then we can delete the lowest importance features and make a scatter plot of the data using the first and second eigenvenctor.

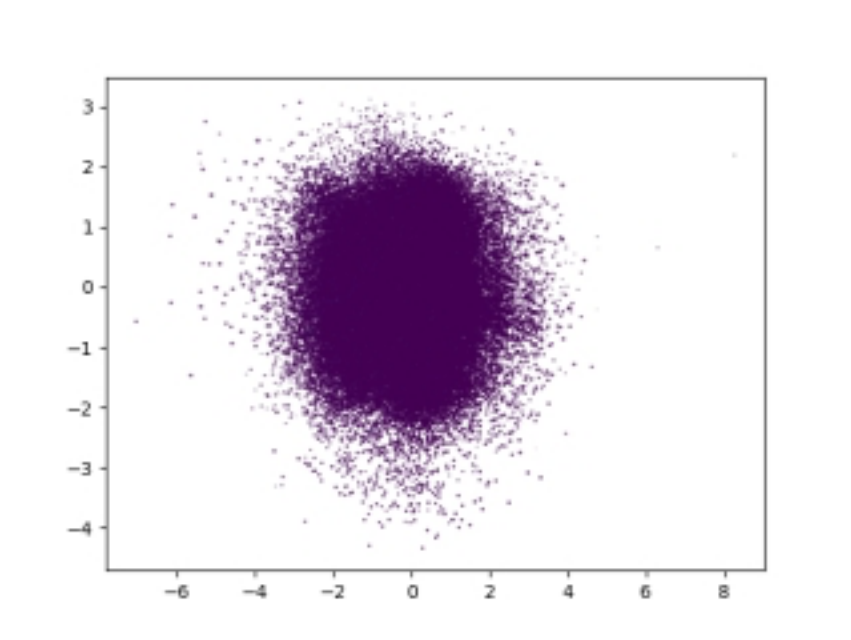


The three color regions in the figure are made using the first and second eigenvenctor of the PCA results, representing accidents that occur in different latitude and longitude regions, respectively. The two right-point data can be seen as the wrong data, so delete it.

Third, Use LDA data to reduce dimensionality.(By LDA.py)

```
url1 = 'caracteristics.csv'
df = pd.read_csv(url1,encoding="LATIN_1",low_memory=False)#iso8859_15
X = df
X_train=X[['lum', 'int', 'atm','col','lat','long', 'catr', 'circ', 'nbv', 'vosp','prof', 'plan', 'lartpc', 'larrout', 'surf', 'infra', 'situ', 'env1']]
Y_train=X[['unscathed', 'killed', 'hospital', 'light']]
fig = plt.figure('LDA')
lda = LinearDiscriminantAnalysis(n_components=3)
lda.fit(X_train,Y_train['killed'])
X_new = lda.transform(X_train)
print("The ratio of the variance value to the total variance of each principal component after dimensionality reduction:",lda.explained_variance_ratio_)
print("Sample size and dimensions before dimension reduction:",X_train.shape)
print("Sample size and dimensions after dimensionality reduction:",X_new.shape)
plt.scatter(X_new[:, 0],X_new[:, 1],X_new[:, 2],marker='o',c=Y_train['killed'],alpha=0.5)

plt.show()
```

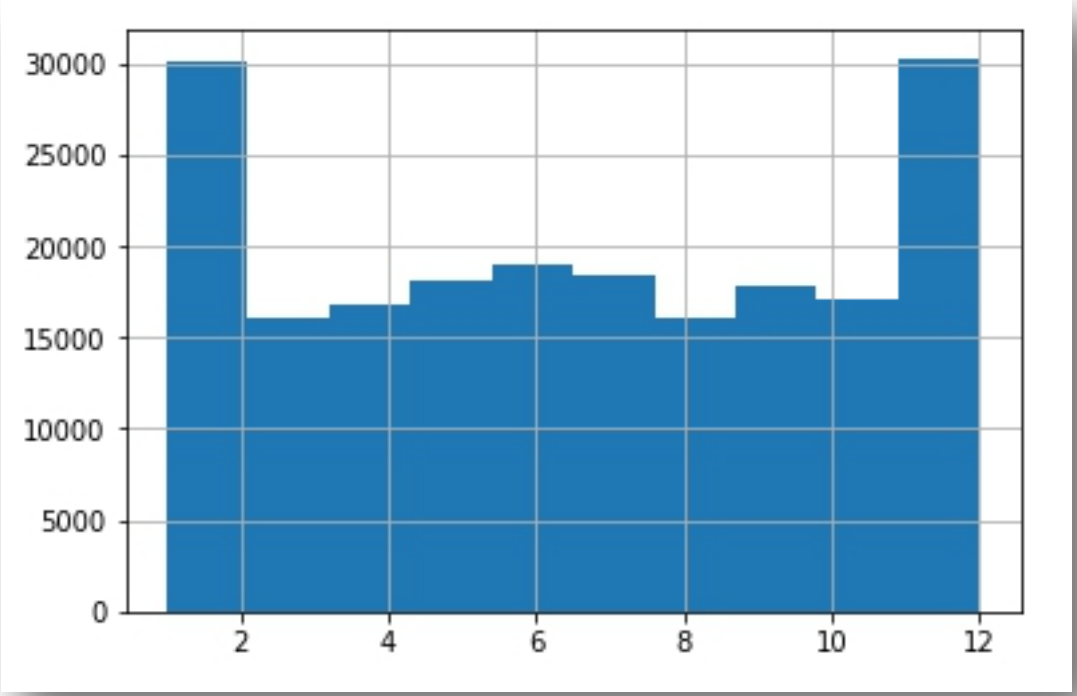


iv. Analysis of the dataset

Calculate the option that has the most occurrences of each column in the 'caracteristics.csv' by 'calculate.py':

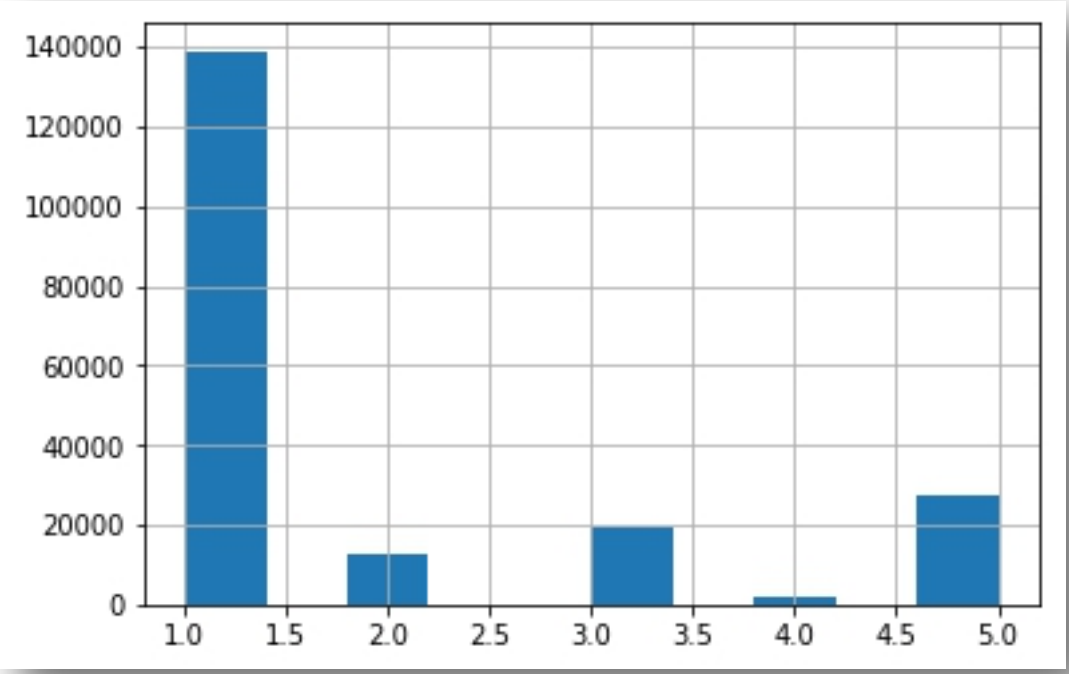
```
an
7                8815
8                10753
10                22864
11                22323
12                20162
13                20041
14                22163
15                29016
16                40652
Max:                16                40652
mois
1                15947
2                13793
3                15887
4                16586
5                17803
6                18736
7                18099
8                15703
9                17587
```

10	16900	
11	15226	
12	14522	
Max:	6	18736



This picture shows the number of accidents in each month. It can be seen that the month from November to the second year is the peak of the accident, and the number of accidents is nearly twice that of other months.

lum		
1	136709	
2	12370	
3	19239	
4	1574	
5	26897	
Max:	1	136709



This picture depicts the number of accidents under different lighting conditions.

1 - Full day

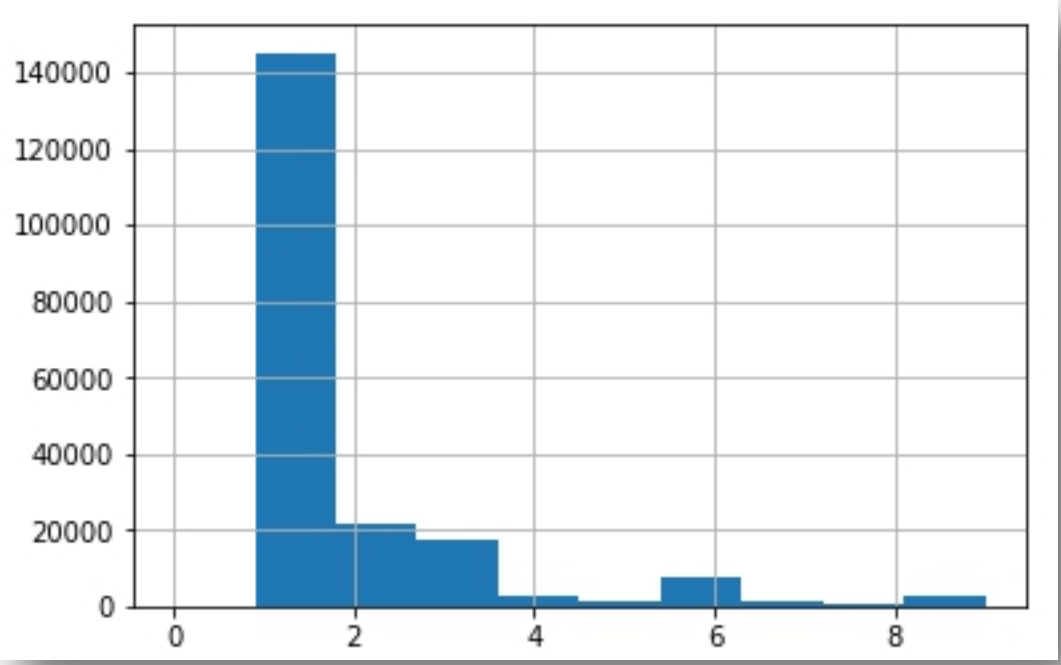
2 - Twilight or dawn

3 - Night without public lighting

4 - Night with public lighting not lit

5 - Night with public lighting on

	int	
0	5	
1	142774	
2	21565	
3	16971	
4	2636	
5	1228	
6	7419	
7	1324	
8	232	
9	2635	
Max:	1	142774



This picture depicts the number of accidents that occurred in different intersection types.

1 - Out of intersection

2 - Intersection in X

3 - Intersection in T

4 - Intersection in Y

5 - Intersection with more than 4 branches

6 - Gyratory

7 - Place

8 - Level crossing

9 - Other intersection

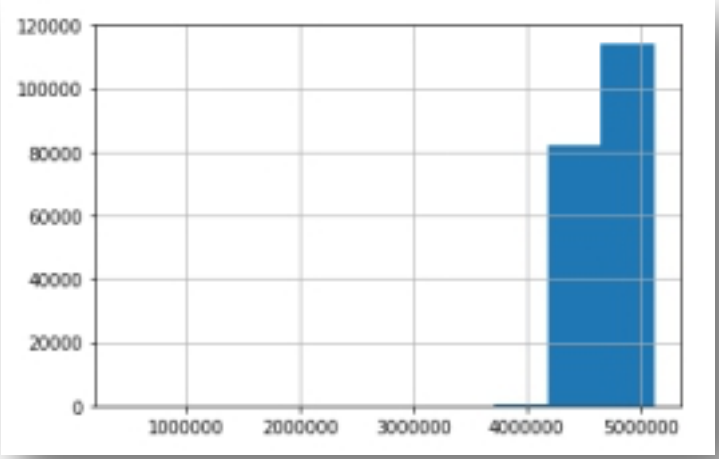
atm	
0	3
1	158527
2	19544
3	4644
4	1152
5	1751
6	561
7	2672
8	6607
9	1328

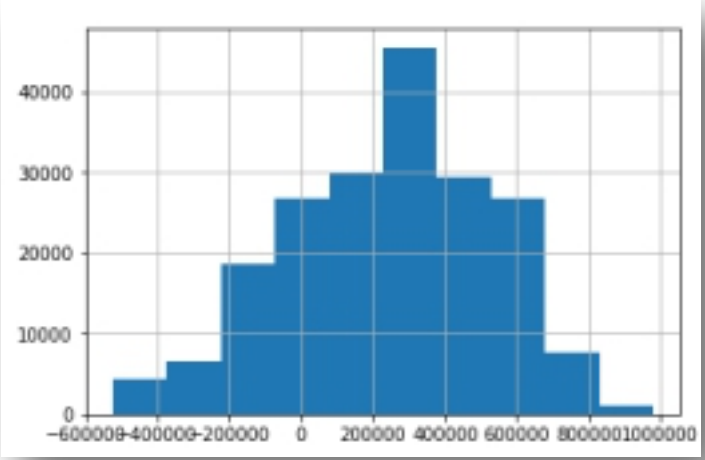
Max:	1	158527
col		
1	24266	
2	20939	
3	53766	
4	5366	
5	6718	
6	63840	
7	21894	
Max:	6	63840
catr		
1	13421	
2	11078	
3	86281	
4	80901	
5	330	
6	1508	
9	3270	
Max:	3	86281
circ		
0	11675	
1	24498	
2	136278	
3	23354	
4	984	
Max:	2	136278
vosp		
0	186031	
1	3613	
2	2775	
3	4370	
Max:	0	186031
prof		
0	17117	
1	143090	
2	29636	
3	4009	
4	2937	
Max:	1	143090
plan		
0	12907	
1	143822	
2	19579	
3	17445	

```
4                   3036
Max:                   1                143822
lartpc
0                   169232
1                   61
2                   70
3                   68
4                   68
5                   498
6                   185
7                   142
8                   263
9                   89
10                   3177
11                   140
12                   263
13                   118
14                   108
15                   5902
...
800                   5
811                   1
814                   1
840                   1
900                   1
907                   1
915                   1
Max:                   0                169232
larrout
0                   67266
1                   18
2                   2
3                   8
4                   27
5                   43
6                   118
7                   77
8                   27
9                   13
10                   162
11                   9
12                   17
13                   12
14                   12
```

```
15                  70
16                  3
17                  11
18                  11
19                  5
20                  247
...
950                 1
960                 2
970                 1
990                 3
999                 2
Max:                0                   67266
surf
0                   7362
1                   151605
2                   33814
3                   225
4                   68
5                   589
6                   148
7                   1342
8                   453
9                   1183
Max:                1                   151605
infra
0                   173874
1                   1369
2                   2679
3                   2608
4                   650
5                   14146
6                   1336
7                   127
Max:                0                   173874
situ
0                   11447
1                   163444
2                   1105
3                   16169
4                   3612
5                   1012
Max:                1                   163444
env1
```

0	101521	
3	9705	
99	85563	
Max:	0	101521





Lat Long

The above two figures show the aggregation of different latitude and longitude accidents.The true value of latitude and longitude needs to shift the decimal point of the coordinates in the graph to the left by 5 digits.We can see that more accidents occurred between 2-4 degrees east longitude.Then the accidents on both sides of the area are gradually reduced

In addition to latitude and longitude, the attributes in our data set are discrete rather than continuous.

We use 'start.py' to build different machine learning models to predict the number of different degrees of injuries caused by accidents that have occurred.

Here is graph of the accuracy of predicting the number of deaths and the number of predicted deaths using four models.

```
#0.8570357035703571/0 ⟶ #0.8570357035703571/0 ⟶ ⟶ Neural Networks
#0.7875787578757876/233 #0.741024102410241/519 ⟶ ⟶ Decision tree
#0.8362336233623362/84 ⟶ #0.8379837983798379/67 ⟶ ⟶ KNN
#0.8563856385638564/3 ⟶ #0.8517851785178517/38 ⟶ ⟶ Random Forest
```

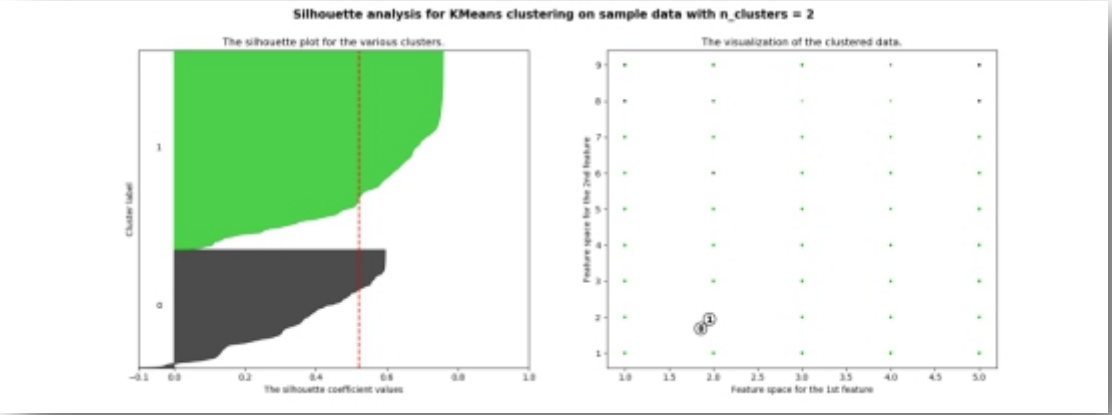
Clusting(by clusting.py)

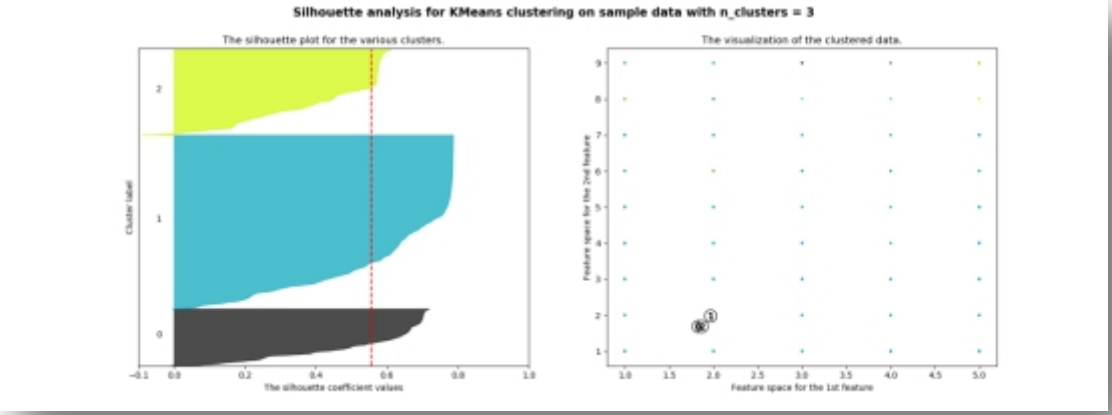
Silhuette Analysis(by SilhuetteAnalysis.py)

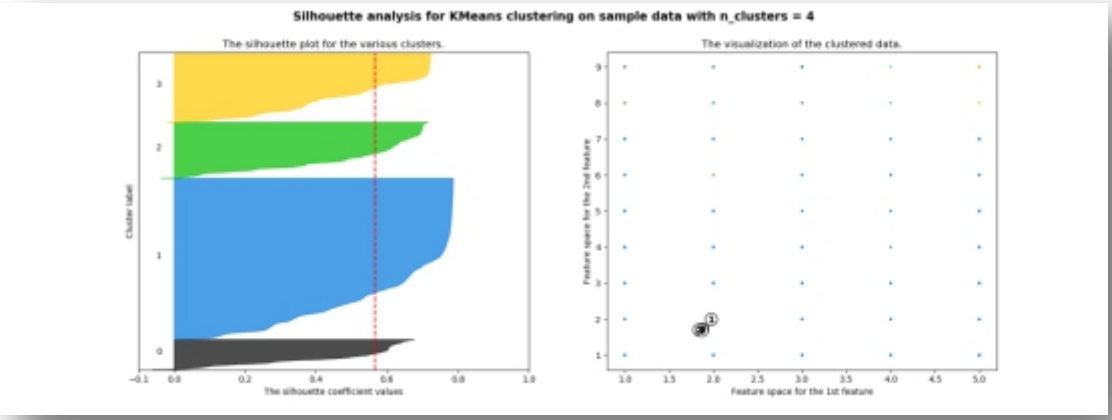
When n=2, the size of the first cluster is much larger than the 0th cluster.

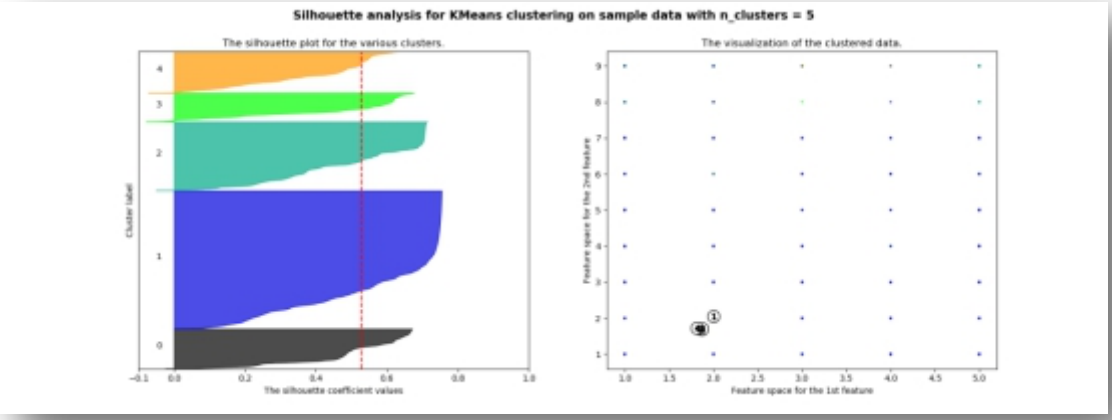
When n=4, 5, 6, there is always a cluster size that is much larger than other cluster sizes, and the distribution is not uniform, so clustering results are not applicable when n_clusters is these values.

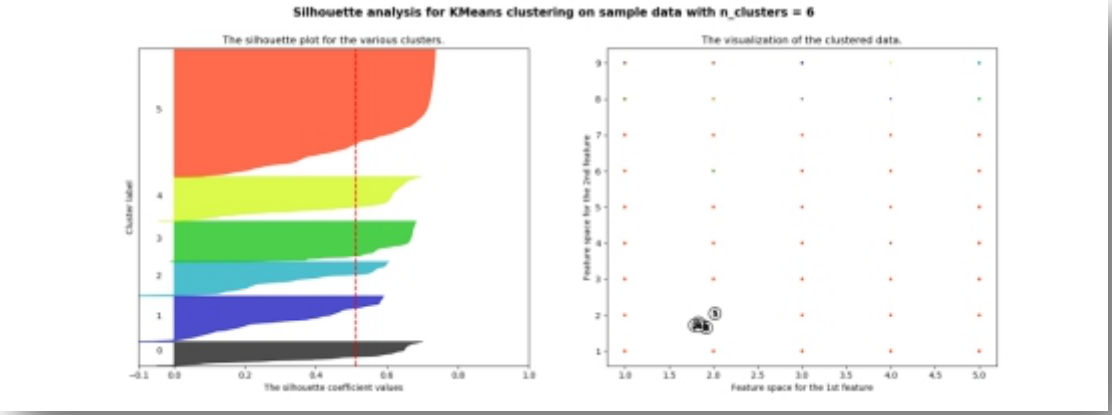
When n=3, the overall clustering effect of the three types of clustering is still larger than the other two categories, but I think the three types are closer in size, so use n=3 for clustering.




Silhouette analysis for KMeans clustering on sample data with n_clusters = 3
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 4
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature


Silhouette analysis for KMeans clustering on sample data with n_clusters = 5
The silhouette plot for the various clusters.
The visualization of the clustered data.
Cluster label
The silhouette coefficient values
Feature space for the 2nd feature
Feature space for the 1st feature



I don't think these clusters have a particularly good effect. But I think when n_clusters=3 is best now.

5 Construction step (Step III)

i. Integration of all the cases defined in the elaboration step.

We have two predicting features and five data visualization features.

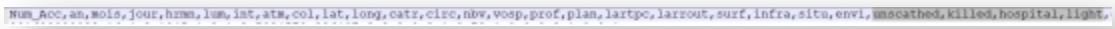
1.Predict the severity of the accident (under the premise of an accident)

A.Use the already processed data to build a machine learning model for training.

In this function, we are prepared to predict the number of minor injuries, deaths, hospitalizations, and injuries not injured in the accident by user-entered conditions.

First, add four columns of attributes to caracteristics.csv, which are unscathed,killed,hospital and light. The functions are called users.csv to calculate the number of people in each incident.Then divide the data set into two parts: training data and test data.

```
for index in range(len(df)):
    ban = df.loc[index]
    df2=df1[(df1.week == ban.week)&(df1.time == ban.time)]
    df2 = df2.reset_index()
    for i in range(len(df2)):
        df.loc[index,'unscathed'] = df.loc[index,'unscathed'] + df2.loc[i,'unscathed']
        df.loc[index,'killed'] = df.loc[index,'killed'] + df2.loc[i,'killed']
        df.loc[index,'hospital'] = df.loc[index,'hospital'] + df2.loc[i,'hospital']
        df.loc[index,'light'] = df.loc[index,'light'] + df2.loc[i,'light']
    print(index)
'''
```



Second, Machine learning building model

Specific details will be shown in the Machine Learning step.

B.Provide interface

Read the model content through functions, and provide result data for the front-end display prediction effect.

```
def predictDeaths(accident):#accident对象的数组
    MLD = joblib.load('Dtree.model')
    df = pd.DataFrame(columns=('Num_Acc','an','mois','jour','hrmn','lum','int','atm','col','lat','long','catr','circ'
    for index in range(len(accident)):
        df.loc[index] = accident[index].tolist()
    X_test = df[['lum','int','atm','col','catr','circ','nbv','vosp','prof','plan','lartpc','larrout','surf','inf
    print(X_test)
    P = MLD.predict(X_test)#list
    return P
```

i. Predict the probability of an accident occurring under specified conditions.

A.Use the already processed data to build a machine learning model for training.

In this function,We are prepared to predict the possibility of an accident under certain conditions in France.

First, Add the attribute chance and set the chance value of the existing incident data to 1.Randomly generate fake traffic accident data and mark their chance value as 0

Second, Machine learning building model

Specific details will be shown in the Machine Learning step.

B.Provide interface

Read the model content through functions, and provide result data for the front-end display prediction effect.

```
def predictTheProbability():#accident对象的数组
    df = pd.read_csv(predictChance,encoding="LATIN_1",low_memory=False)#iso8859_15
    MLP = joblib.load('Regression.model')
    X_test = df[['lum','int','atm','col','catr','circ','nbv','vosp','prof','plan'
    Z = MLP.predict(X_test)#list
    for i in range(len(Z)):
        if Z[i] < 0:
            Z[i] = 0
        elif Z[i] > 1:
            Z[i] = 1
    X_position = df[['lat','long','chance']]
    X_position['chance'] = pd.DataFrame(Z,columns=['chance'])
    return X_position.values.tolist()
```

ii. Machine Learning

Predict the severity of the accident

I use the already processed data to train the model.

Using data items ['lum', 'int', 'atm','col', 'catr', 'circ', 'nbv', 'vosp','prof', 'plan', 'lartpc', 'larrout', 'surf', 'infra', 'situ', 'env1'] as input, the number of deaths, minor injuries, hospitalizations, and uninjured persons are the target values.

Next select the model for training.

The number of people we want to predict is a discrete value, so in the choice of model, we should choose the classification model.

We selected multiple models for training, and selected the optimal model by evaluating the results of the model.

Finally we chose the decision tree as the predictive model.

Here is the code and model evaluation results：

```
def Dtree():
    X = pd.read_csv(url1,encoding="LATIN_1",low_memory=False)#iso8859_15
    XT = pd.read_csv(url11,encoding="LATIN_1",low_memory=False)#iso8859_15
    #X = pd.read_csv(urlP,encoding="LATIN_1",low_memory=False)#iso8859_15
    #XT = pd.read_csv(urlPT,encoding="LATIN_1",low_memory=False)#iso8859_15
    X_train=X[['lum', 'int', 'atm','col', 'catr', 'circ', 'nbv', 'vosp','prof', 'plan', 'lartpc', 'larrout', 'surf', 'infra',
    Y_train=X[['unscathed', 'killed', 'hospital', 'light']]
    X_test=XT[['lum', 'int', 'atm','col', 'catr', 'circ', 'nbv', 'vosp','prof', 'plan', 'lartpc', 'larrout', 'surf', 'infra',
    clf = tree.DecisionTreeClassifier()
    clf.fit(X_train,Y_train)
    #tree.plot_tree(clf.fit(X_train,Y_train))
    #with open("jueceshu.dot", 'w') as f:
    #    f = tree.export_graphviz(clf, out_file = f)
    Z=clf.predict(X_test)
    pd_data = pd.DataFrame(Z,columns=['unscathed','killed','hospital','light'])
    print(pd_data)
    #pd_data.to_csv(urlT,encoding="LATIN_1",index=False)
    #joblib.dump(clf,'Dtree.model')
    print(classification_report(XT['killed'], pd_data['killed']))
```

	precision	recall	f1-score	support
0.0	0.86	0.89	0.87	17139
1.0	0.17	0.11	0.13	2617
2.0	0.02	0.06	0.03	197
3.0	0.00	0.00	0.00	31
4.0	0.00	0.00	0.00	9
5.0	0.00	0.00	0.00	3
6.0	0.00	0.00	0.00	1
7.0	0.00	0.00	0.00	1
8.0	0.00	0.00	0.00	0
10.0	0.00	0.00	0.00	0
26.0	0.00	0.00	0.00	0
micro avg	0.77	0.77	0.77	19998
macro avg	0.10	0.10	0.09	19998
weighted avg	0.76	0.77	0.77	19998

Model prediction accuracy

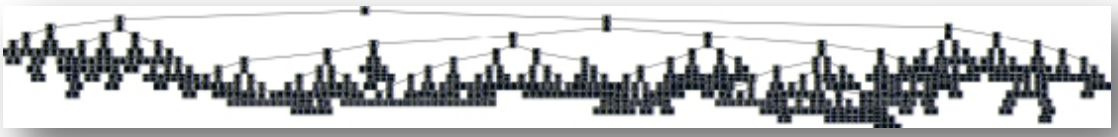
In this picture we can see that MLPClassifier() cannot predict the number of deaths, although he has the highest accuracy rate.

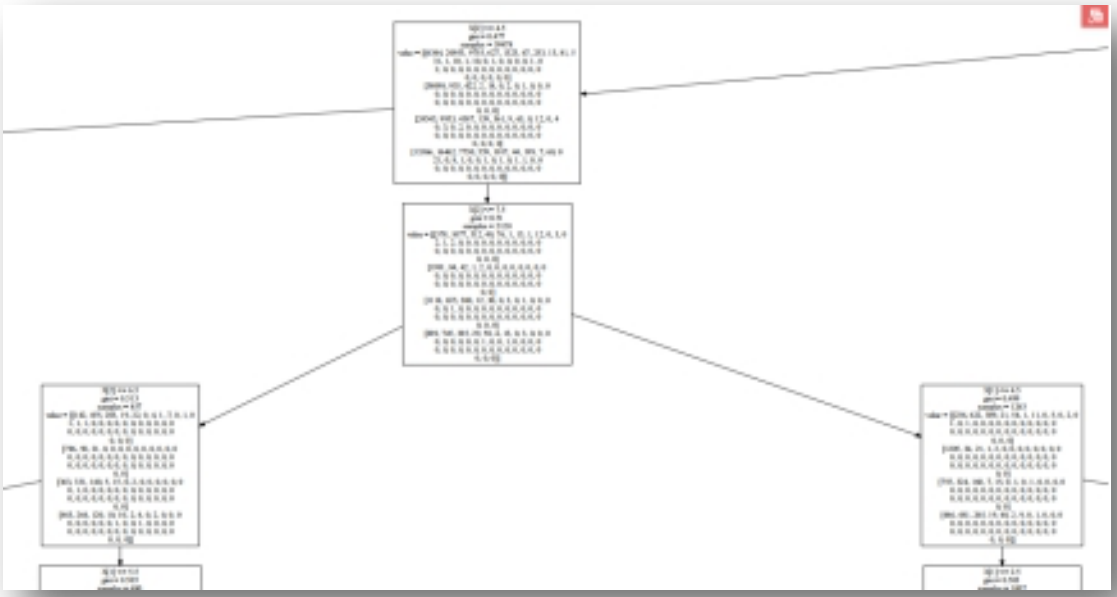
Although the decision tree has a slightly lower accuracy rate, he can better predict the deaths. The number of deaths predicted to be correct by Dtree() is the highest.

```
#0.8570357035703571/0 ⟶ #0.8570357035703571/0 ⟶ ⟶ Neural Networks
#0.7875787578757876/233 #0.741024102410241/519 ⟶ ⟶ Decision tree
#0.8362336233623362/84 ⟶ #0.8379837983798379/67 ⟶ ⟶ KNN
#0.8563856385638564/3 ⟶ #0.8517851785178517/38 ⟶ ⟶ Random Forest
```

Accuracy of different models

Decision tree diagram：





Predict the probability of an accident occurring under specified conditions.

I use the already processed data to train the model.

Using data items ['lum', 'int', 'atm','col', 'catr', 'circ', 'nbv', 'vosp','prof', 'plan', 'lartpc', 'larrout', 'surf', 'infra', 'situ', 'env1'] as input, the chance is the target value.

Next select the model for training.

The number of people we want to predict is a continuous value, so in the choice of model, we should choose the regression model.

I selected multiple models for training, and evaluate model accuracy with MSE.

Finally we chose the SVR as the predictive model.

Here is the code and model evaluation results：

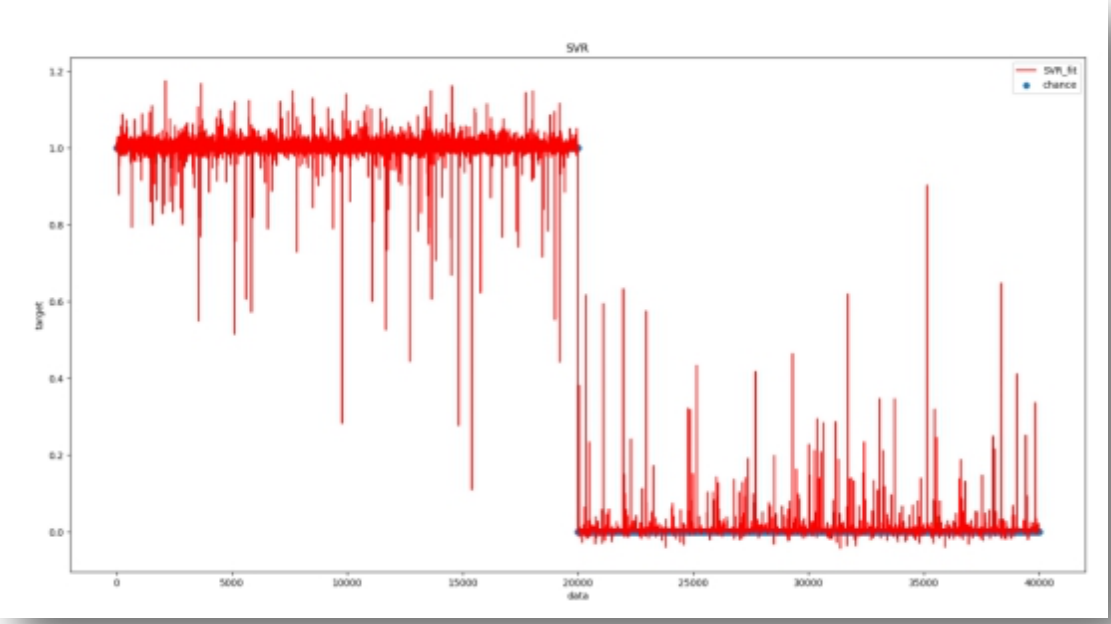
```
def Regression():
    X = pd.read_csv(url1,encoding="LATIN_1",low_memory=False)#iso8859_15
    XT = pd.read_csv(url11,encoding="LATIN_1",low_memory=False)#iso8859_15
    X_train=X[['lum', 'int', 'atm','col', 'catr', 'circ', 'nbv', 'vosp','prof', 'plan', 'lartpc', 'larrout', 'surf'
    Y_train=X['chance']
    X_test=XT[['lum', 'int', 'atm','col', 'catr', 'circ', 'nbv', 'vosp','prof', 'plan', 'lartpc', 'larrout', 'surf'
    Y_test=XT['chance']
    #reg = linear_model.LinearRegression()#0.022112666330325707
    #reg = linear_model.BayesianRidge()#0.022112934511271203
    #reg = svm.SVR(gamma='scale')#0.004849921086918859
    reg = MLPRegressor(activation='logistic')#0.0003363074218161884
    reg.fit(X_train,Y_train)
    #reg = joblib.load('Regression.model')
    Z=reg.predict(X_test)
    pd_data = pd.DataFrame(Z,columns=['chance'])
    print(pd_data)
    pd_data.to_csv(url1T,encoding="LATIN_1",index=False)
    joblib.dump(reg,'Regression.model')
    print(mean_squared_error(Y_test,pd_data))
```

```
0.0003983100455704924
```

MSE Error of SVR:

```
#reg = linear_model.LinearRegression() #0.022112666330325707
#reg = linear_model.BayesianRidge()    #0.022112934511271203
#reg = svm.SVR(gamma='scale')          #0.004849921086918859
reg = MLPRegressor(activation='logistic') #0.0003363074218161884
```

MSE



Model saving and loading

Save the training results as a '*.model' file by using the joblib module.

```
joblib.dump(clf,'Dtree.model')
```

Read and use the model

```
MLD = joblib.load('Dtree.model')
```

iii. Visualization & Data Analysis

(1) Introduction

Visualization is a good way for us to transform the abstract data into vivid graphs, which will help us understand the meaning of our data and even find out some amazing conclusions. For instance, in our project, we combine our data with date and GEO information in order to get some laws underneath the data itself.

We divide our visualization into 3 sections.

The first is to visualize the data we are processing by machine learning. For example, there are some charts to be shown during data cleaning, reducing and clustering.

The second is to visualize the data after clean, classification, such put all the valid data on the maps.

The third is to visualize the data we predict by machine learning.

Pay attention please, all there steps belong to data analysis in this project.

(2) Technology

The first section has been shown before, so all descriptions below are related to secion 2 and 3.

All of the visualization diagrams are based on HTML, CSS, and JavaScript, which means it is easy to build interactive and cross-platform visualization diagrams.

Besides, I use some useful frame to help me, such as Bootstrap, jQuery and Vue.

JSON is the main data format, AJAX is the main method to get data from the data server, which ensure that our data can be updated in real time.

Most importantly, Google Maps API and Baidu Echarts are used as tools to draw the beautiful GUI. Both they provide plentiful methods.

The coding and testing tools I choose are WebStorm (an elegant IDE) , VS Code and Chrome.

(3) Devolopment Environment

MacOS & Linux

(4)Details

①Clustering map with existing traffic accident data

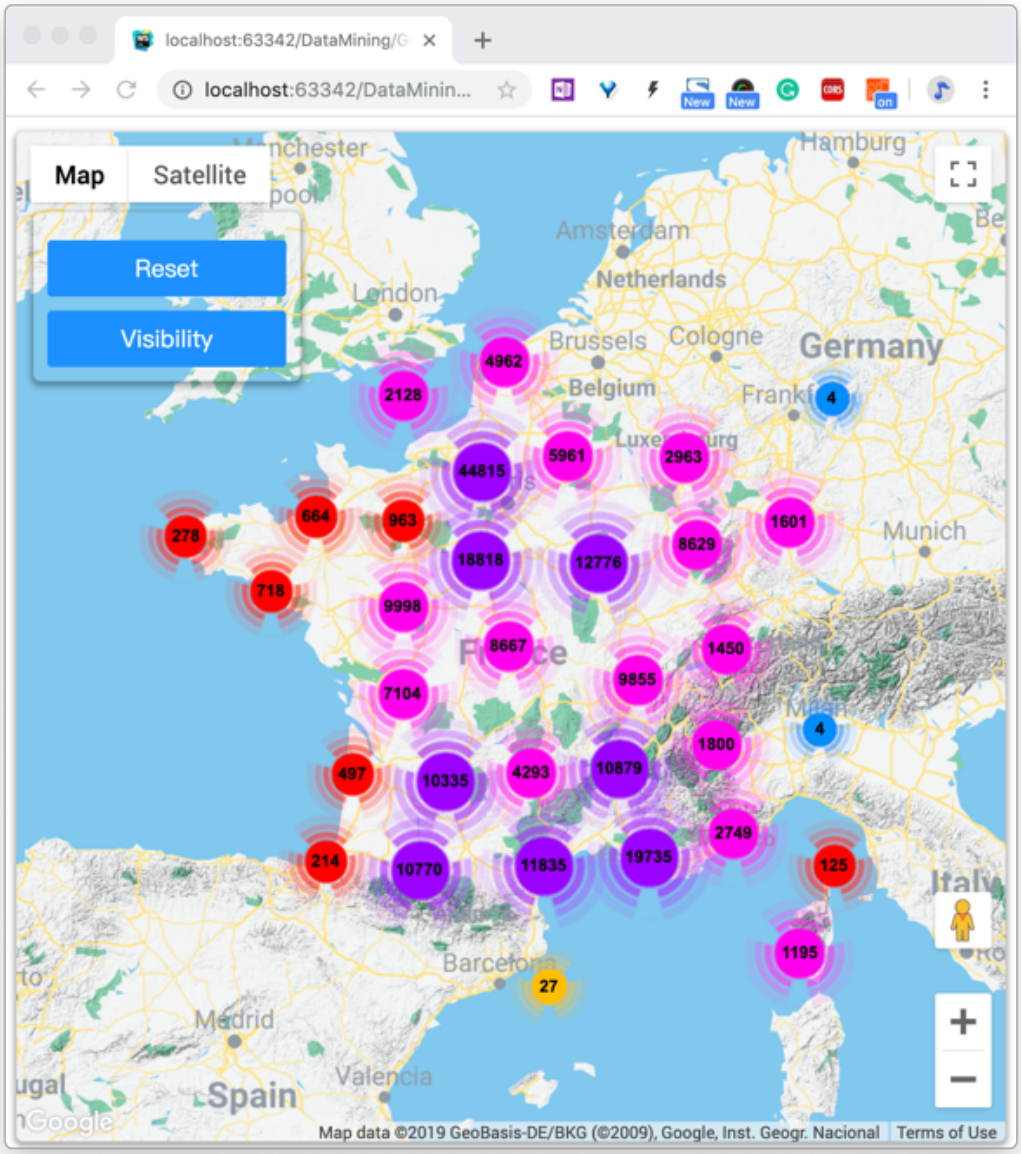
- Introduction and Preview

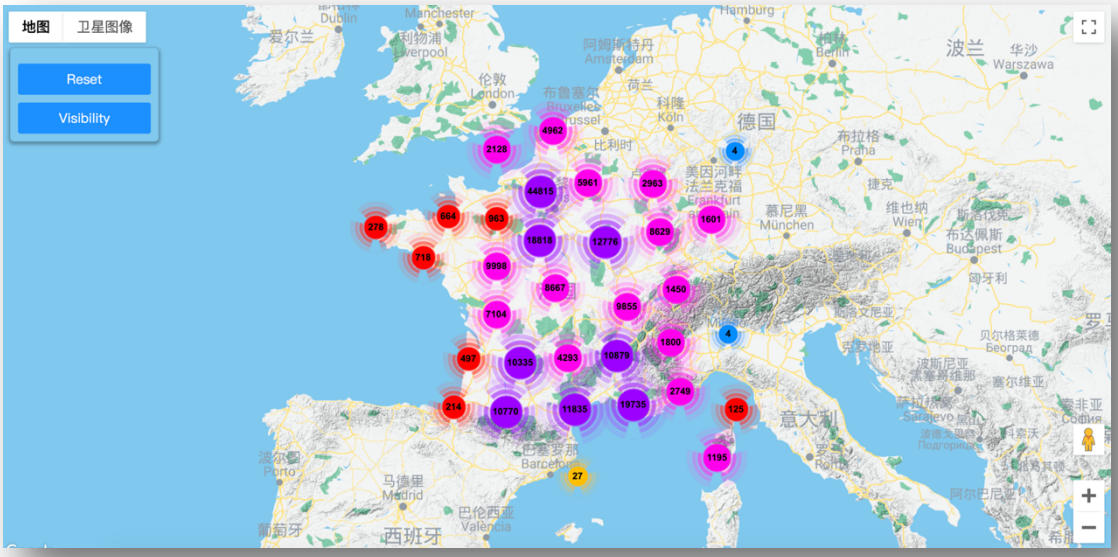
I used Google Maps as a canvas to draw the most effective and useful 200,000 pieces of data from 800,000 pieces of original data, mark them on the map with their

latitude and longitude, and cluster them.

The clustering principle is to calculate the number of markers in a grid of a specified size, and calculate the cluster icon and number of markers in the center.

The cluster icon has five colors and is adjusted according to the size of the data. From high to low, it is purple, pink, red, yellow, and blue.



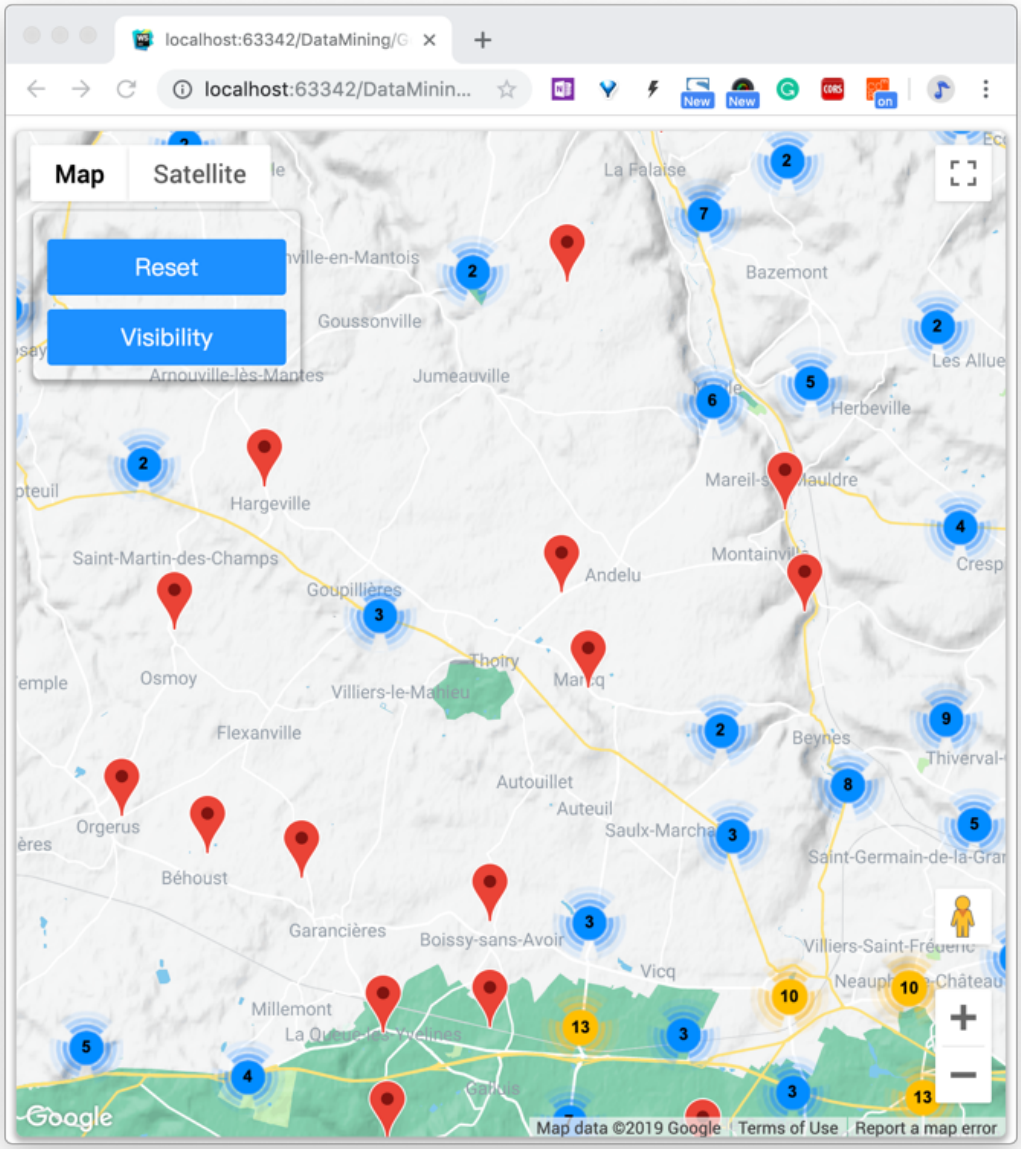


Through clustering, we can observe the number of traffic accidents in various places and the severity of their representatives from a discrete perspective, which helps us to allocate the number of traffic police or staff.

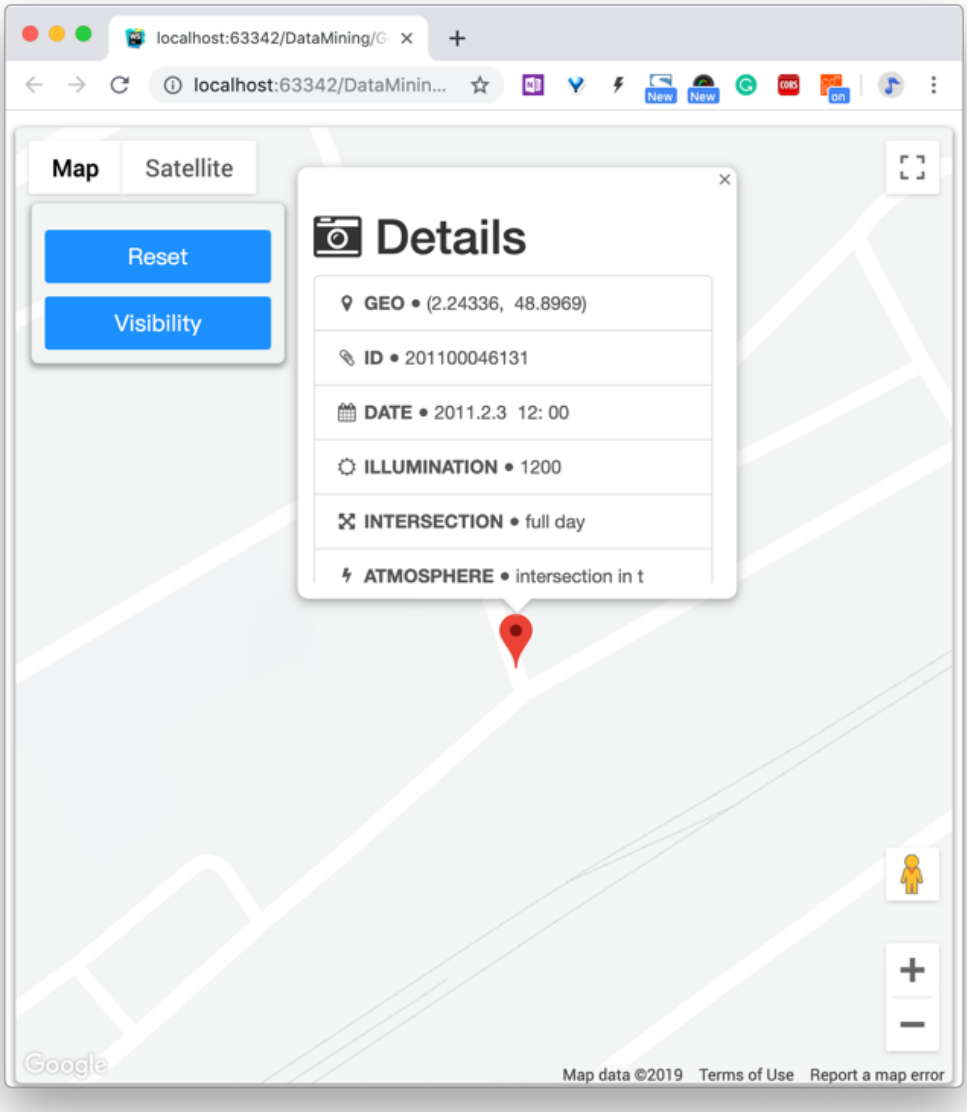
- Interactive:

- Like all normal Google Maps, we can semantically zoom and drag maps with the mouse wheel trackpad or our fingers.

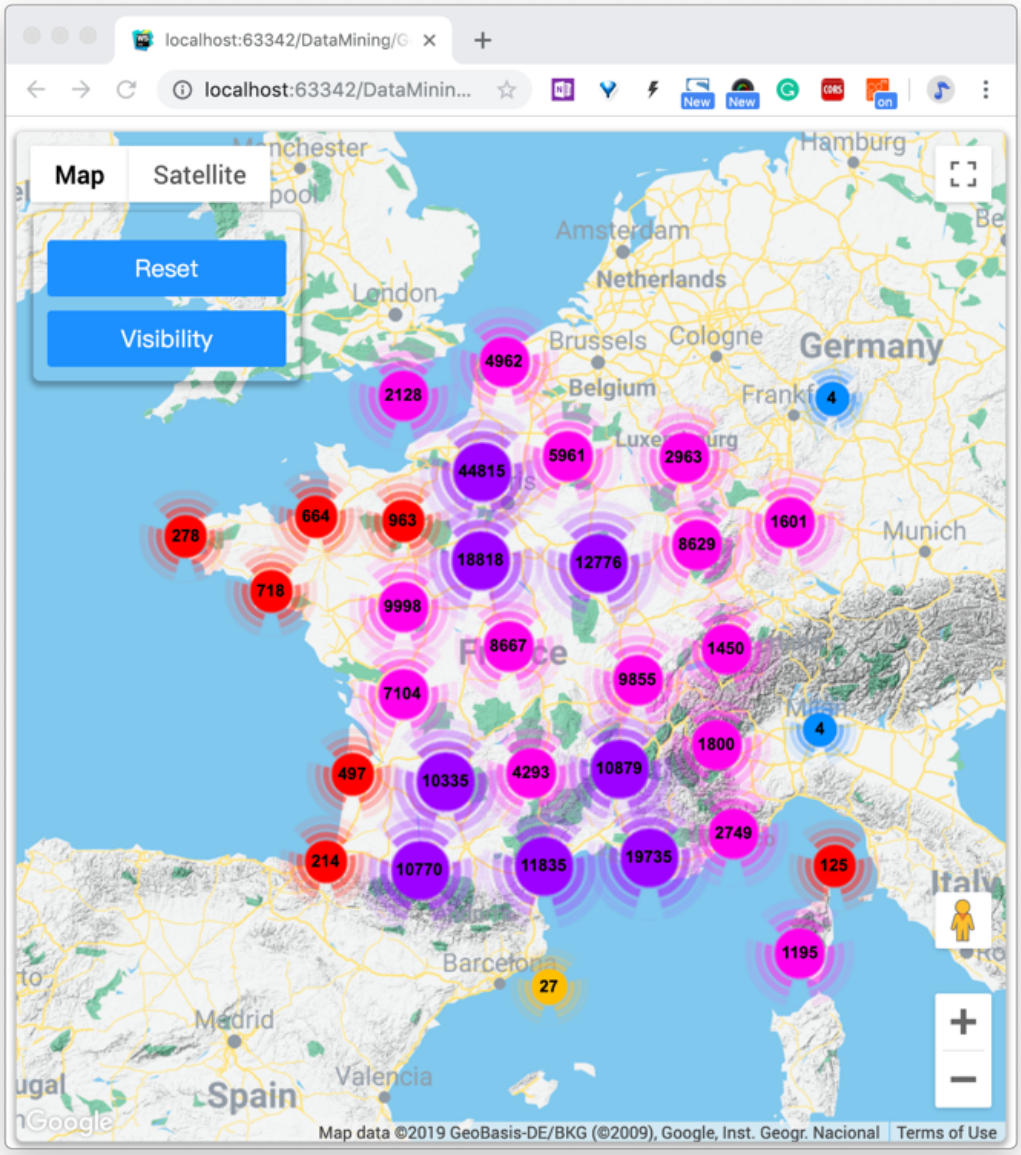
- Click on one cluster icon to enlarge the screen to the grid represented by the cluster icon, and re-cluster the markers in the current range to display the new cluster map.



- Finally we can click on the individual traffic accident marker to open the information window and view the details of the incident.



- We can click the Reset button on the left to return to the initial zoom and position.
- We can click on Visibility button to hide or show the cluster icon



- Analysis

- From these diagrams, we can verify that our data set is very accurate because it can be seen intuitively and clearly from the map that most of the accidents occurred at the intersection and also meet the data in the record.

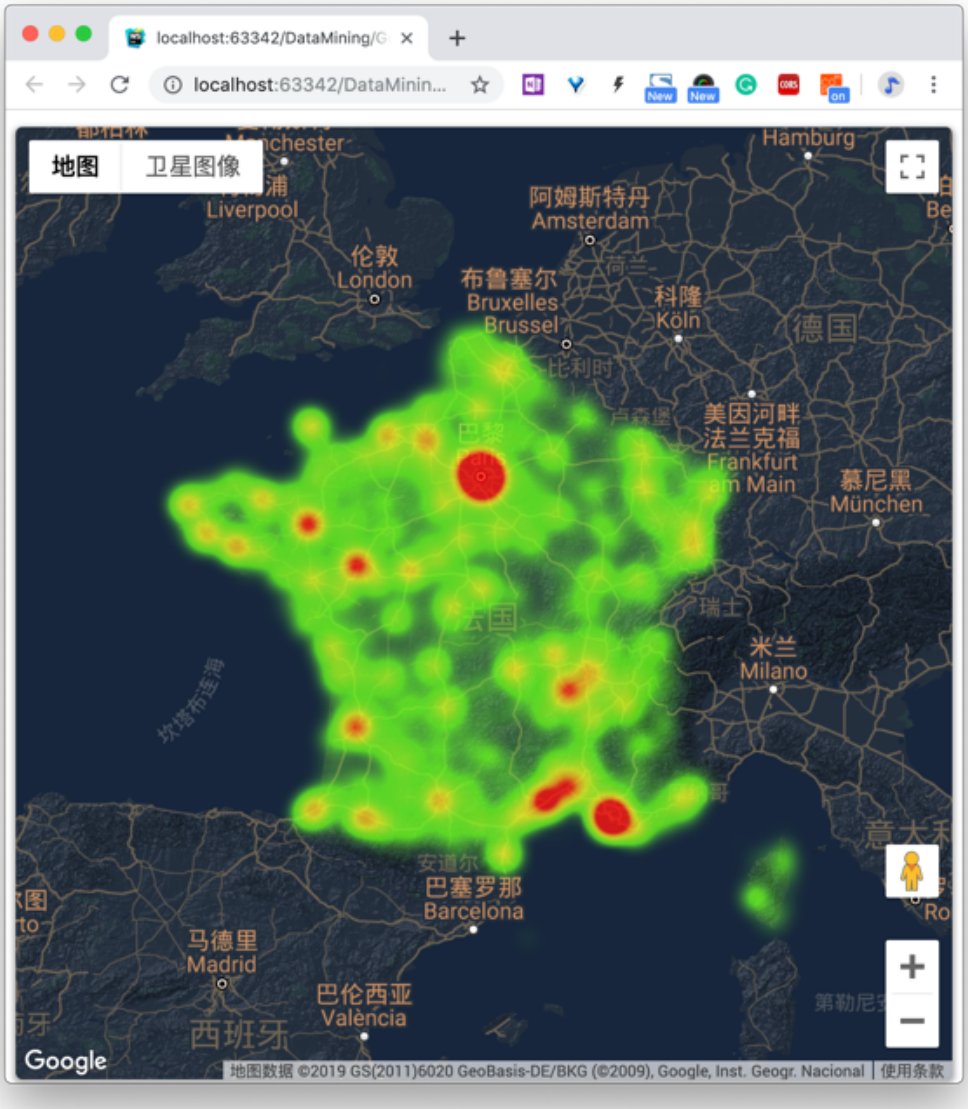
- At the same time, the data itself is basically in line with an abundant of accidents in big cities and a few accidents in suburban counties. Especially in Ile-de-France where we are living now and famous French Riviera in the south of France.

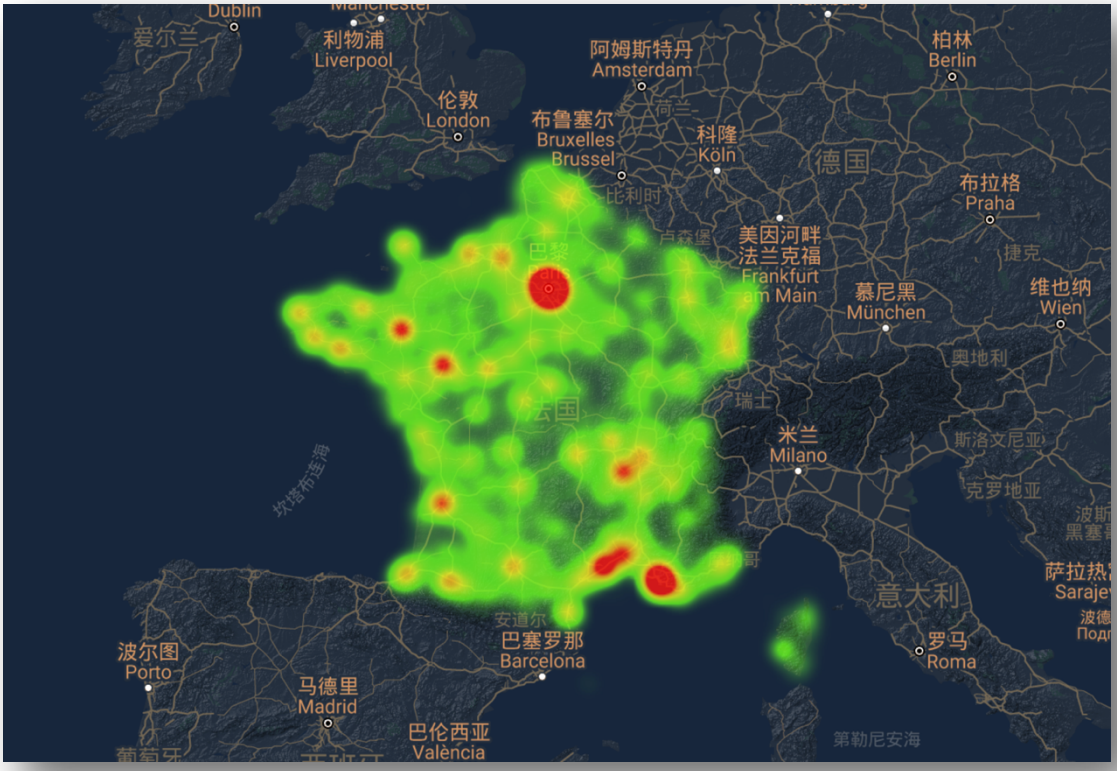
② Heat maps with exsisting accidents

- Introduction and Preview

By converting discrete data points into continuous heat maps, we use the existing data density to roughly simulate the probability of occurrence of a region.

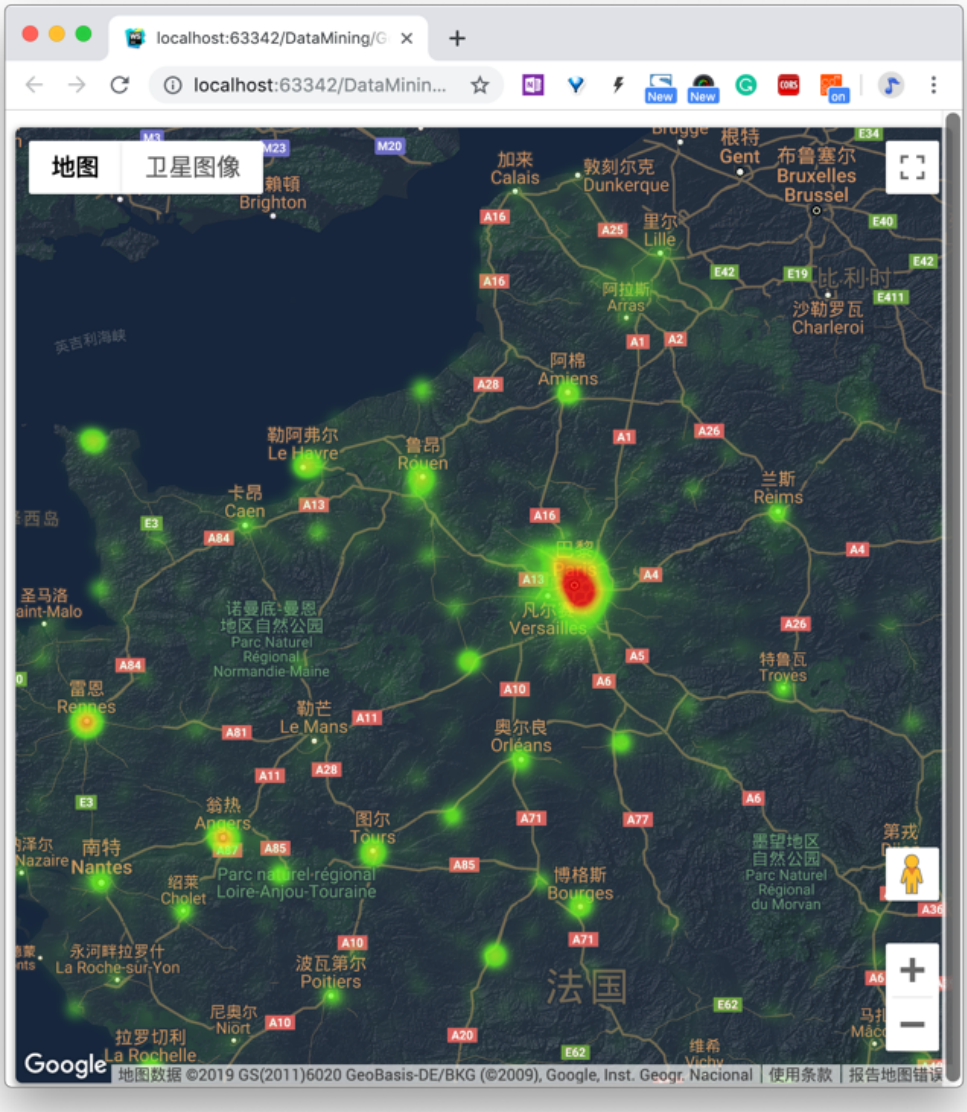
In terms of color depth, red, orange, yellow, and green represent the severity from seriour to light. A custom dark background also helps us see the effects of the heat map.





- Interaction

  - Double-click on an area to zoom in to see the re-rendered heat map for that area.



- Analysis

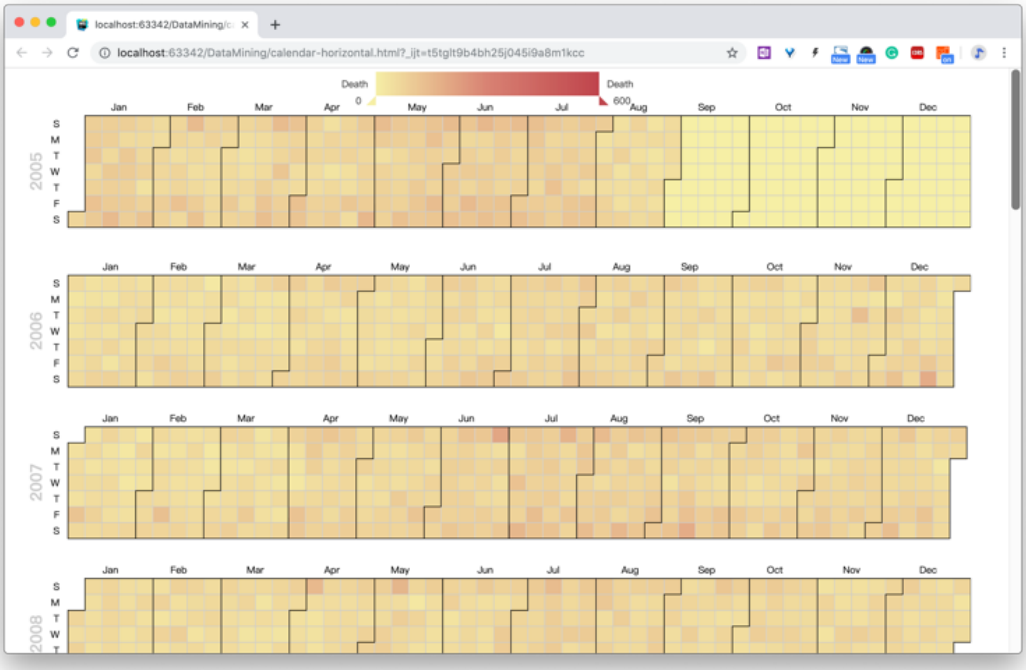
As we can see from the heatmap, we can almost get the same conclusion with the cluster map above. However, heatmap simulate the continuous data while cluster map is discrete. When transportation department add more true data into it, heatmaps will become more and more predictive.

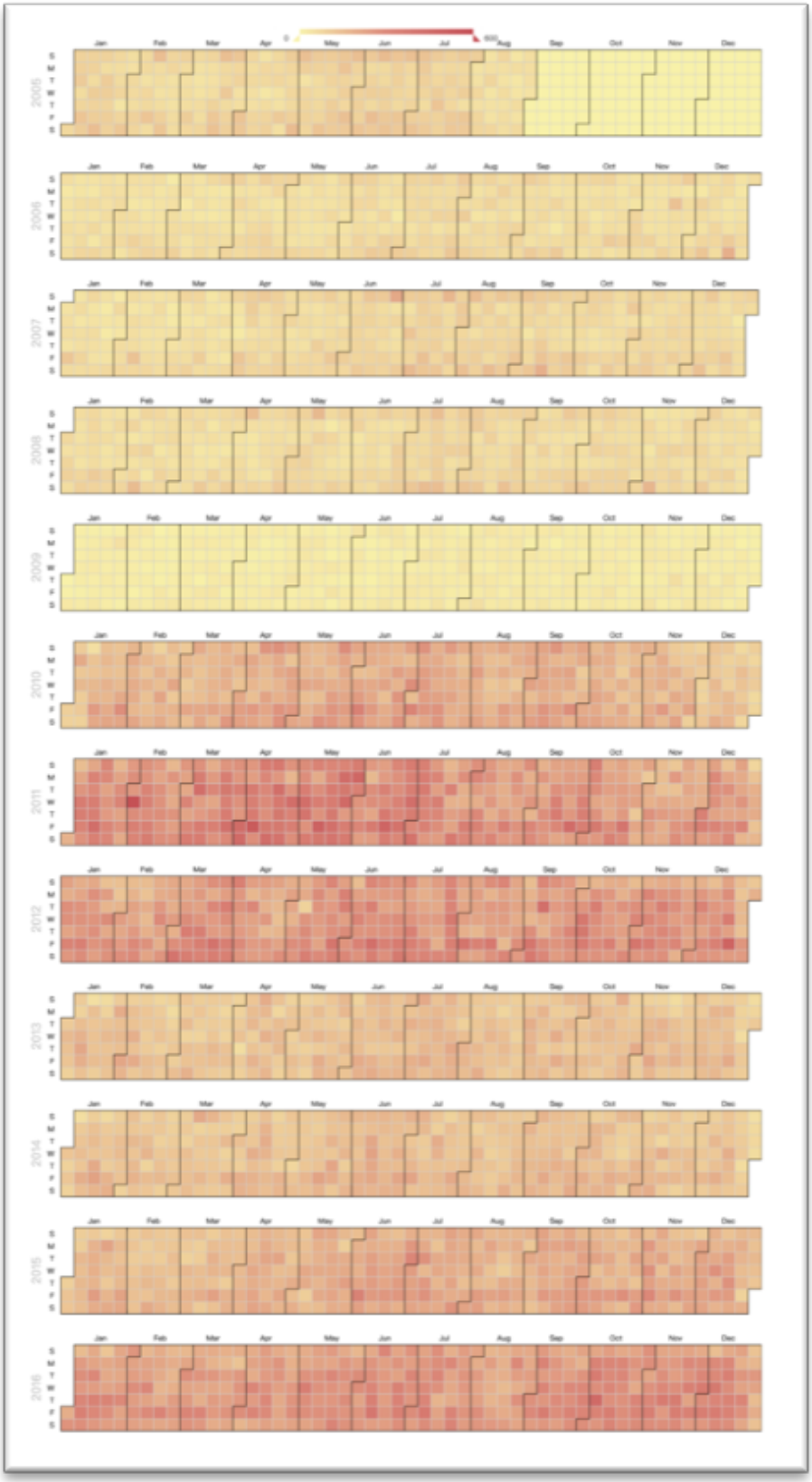
③ Calendar-type heat map of the number of deaths per day for the whole year of 2005-2016

- Introduction

In this diagram, I put all the years together. Each year holds a large grid, and is divided into weeks and months by the little grid. The color depth of each grid indicates the number of deaths, from yellow to red. At the top of the diagram there is a visual

mapping control.

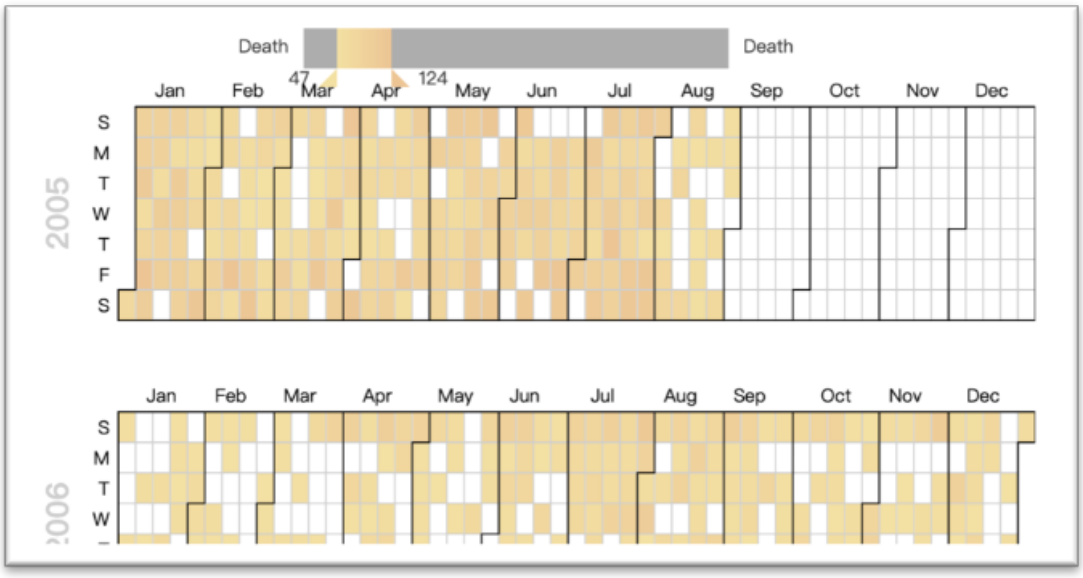



0
600
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
S M T W T F S
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016

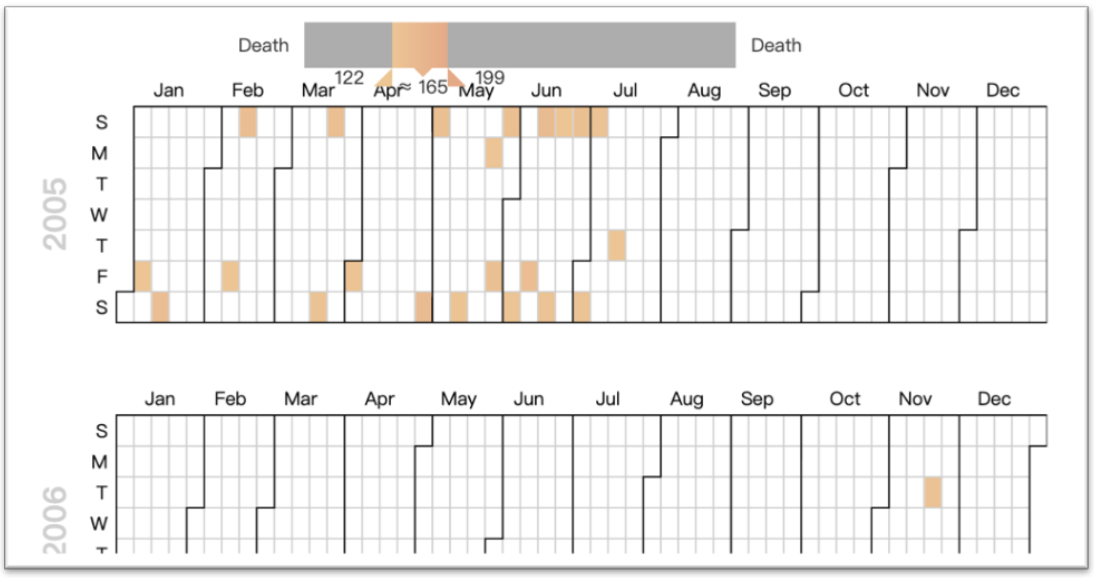
- Interaction

- When the mouse moves over the map control, the square of the value represented by the mouse position is highlighted.

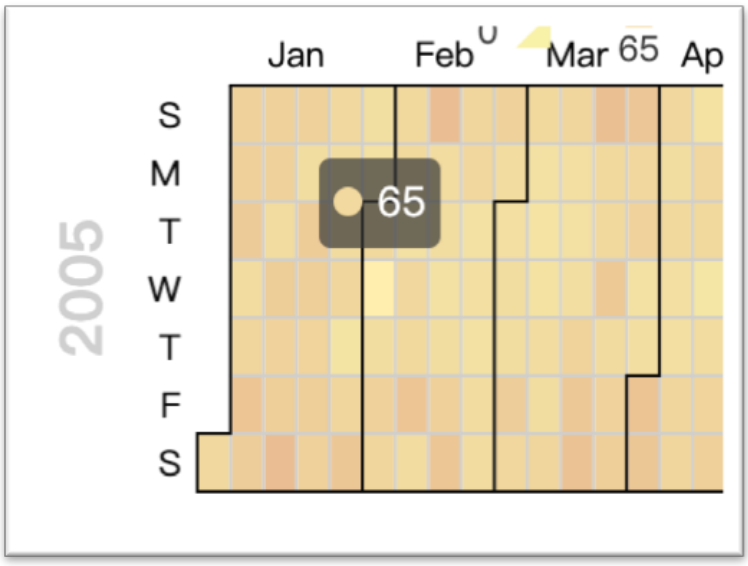
- The mouse can drag the left and right range of the mapping control. The default is all (0 - 600 people). After dragging, only the color blocks of the specified range will be displayed.



- After specifying the range, you can drag this range to change the selected value.



- When we move the mouse over a color block, the color block will be highlighted and the number of deaths represented by the color block will be displayed above



- Analysis

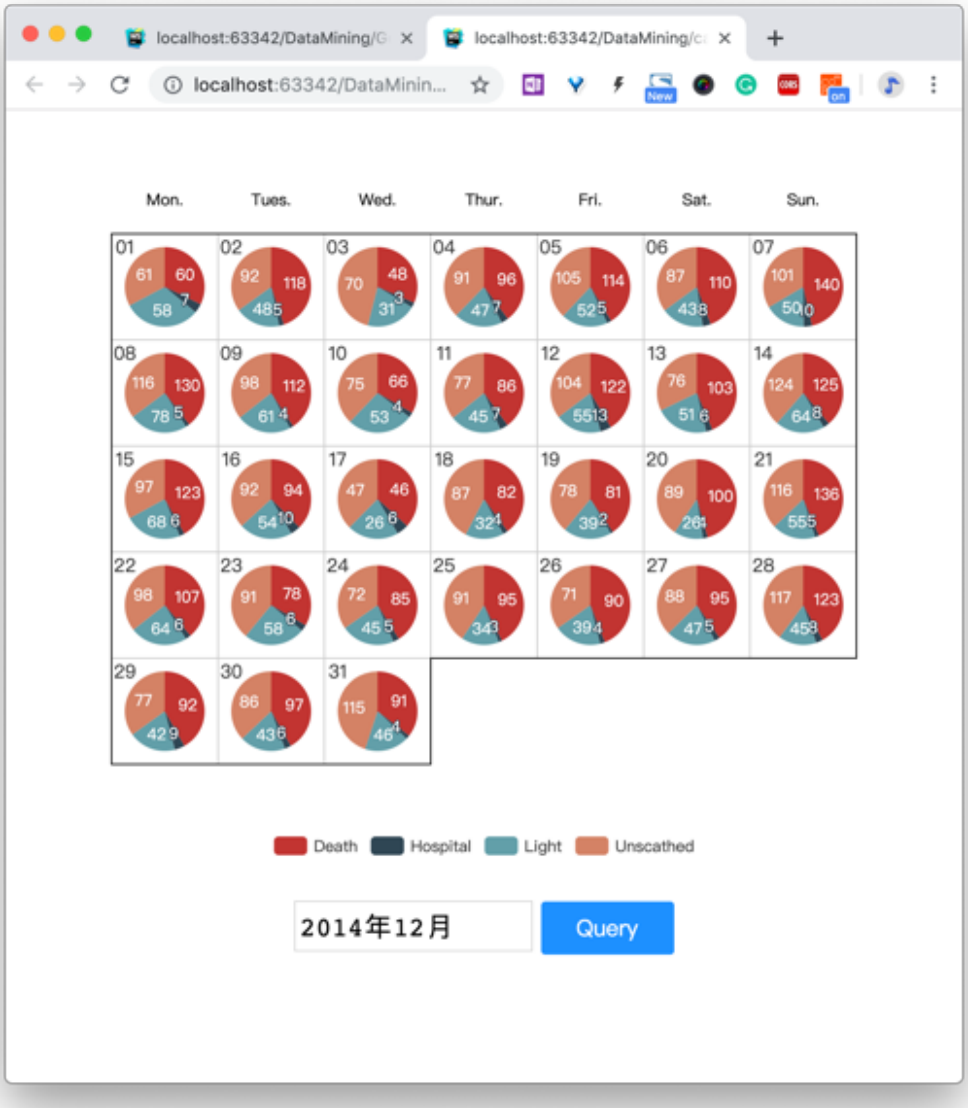
We can see that the number of traffic accidents in France. Obviously, before 2009 it was small, and it has increased significantly since 2010.

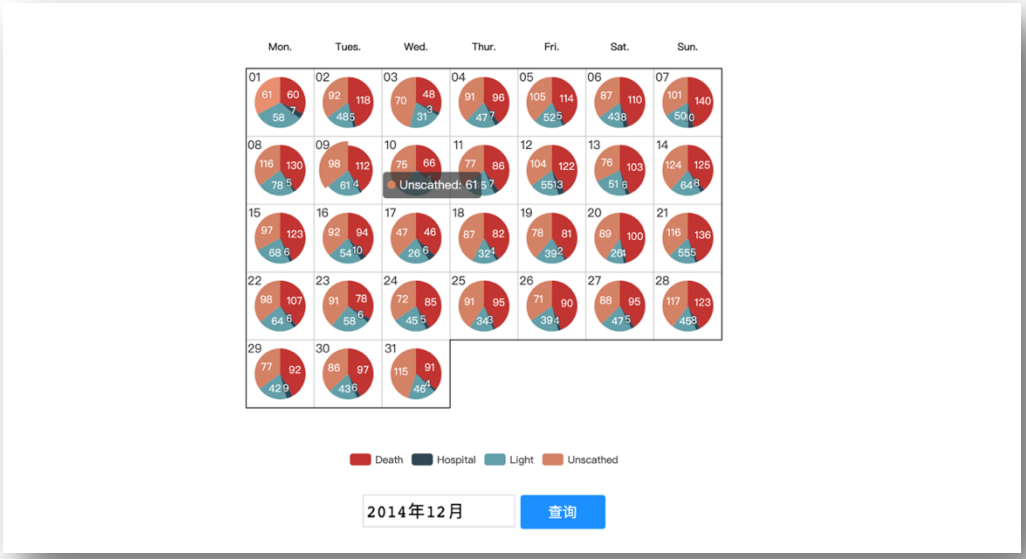
④ Calendar-pie diagram with Inquiring about the number of traffic casualties per day of the month by month

- Introduction and Interaction:

- The center of the graph shows the monthly calendar for the specified month. There is a pie chart in each day. There are four categories in pie charts: death, hospitalization (hospital), minor injury (light), no injury (unscathed), which are expressed in four different colors.

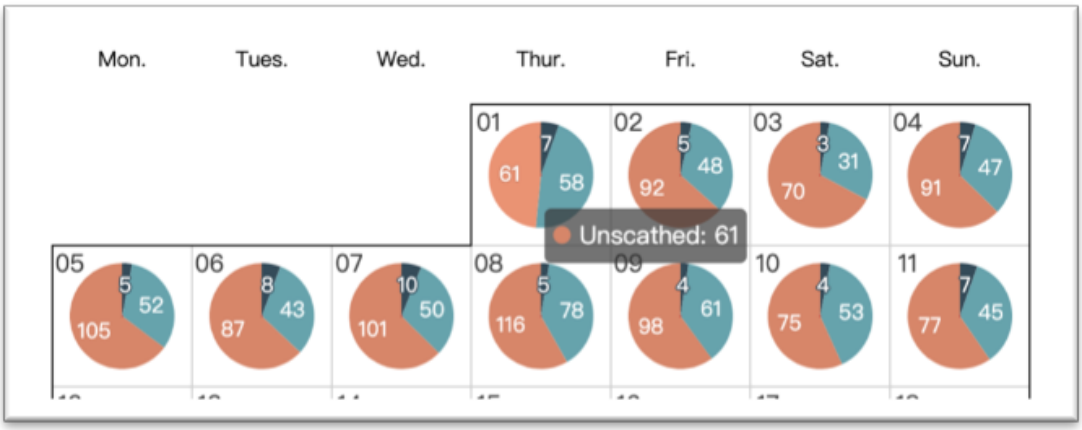
- There is a category selector below, you can click to hide or display a certain type of pie chart , freely combine categories or display only one category.


localhost:63342/DataMining/
localhost:63342/DataMining/
localhost:63342/DataMinin...
Mon.
Tues.
Wed.
Thur.
Fri.
Sat.
Sun.
01 61 60 58 7
02 92 118 485
03 70 48 31 3
04 91 96 47 7
05 105 114 525
06 87 110 438
07 101 140 50 0
08 116 130 78 5
09 98 112 61 4
10 75 66 53 4
11 77 86 45 7
12 104 122 551 3
13 76 103 51 6
14 124 125 64 8
15 97 123 68 6
16 92 94 54 10
17 47 46 26 6
18 87 82 32 1
19 78 81 39 2
20 89 100 26 1
21 116 136 555
22 98 107 64 6
23 91 78 58 6
24 72 85 45 5
25 91 95 34 3
26 71 90 39 4
27 88 95 47 5
28 117 123 45 8
29 77 92 42 9
30 86 97 43 6
31 115 91 46 4
Death
Hospital
Light
Unscathed
2014年12月
Query



- When the mouse is placed on any arc, the type value of the arc is displayed.

- Click on the calendar control below to select the month you want to query, and click the "Query" button to find the related data.



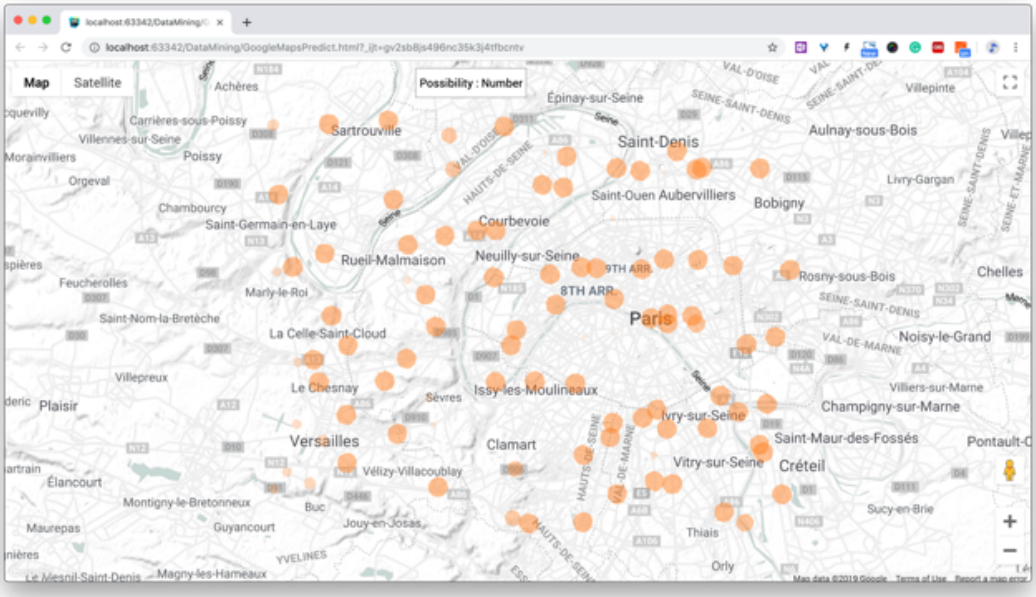
⑤ The Real-time prediction　Map of the probability of traffic accident

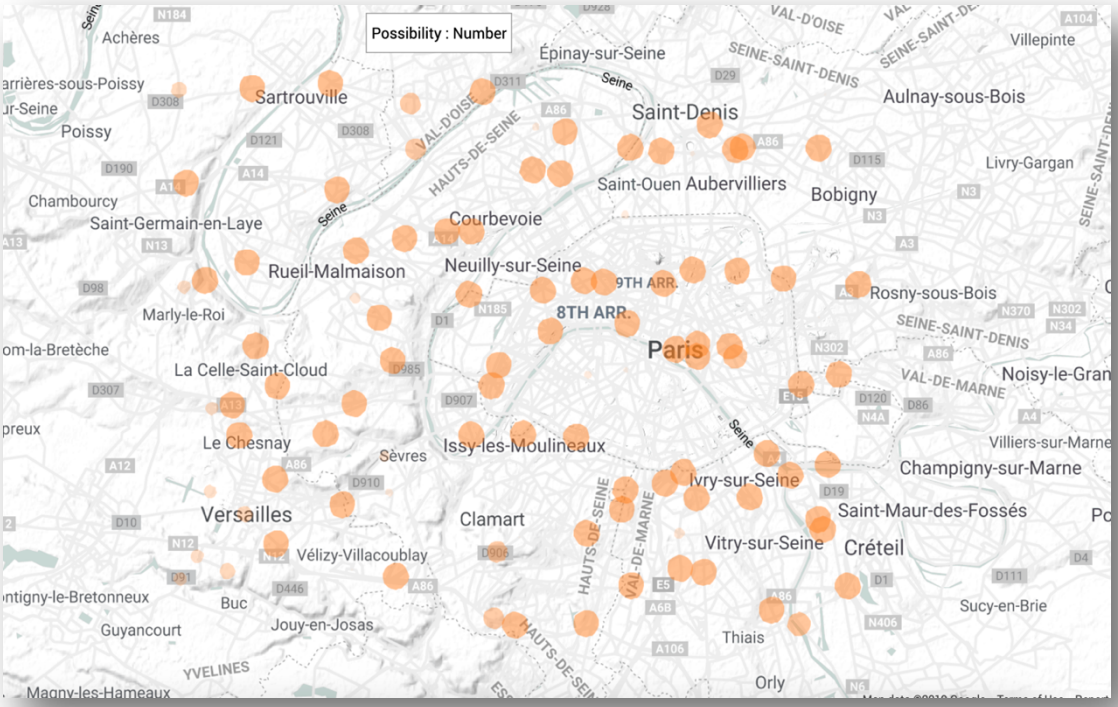
- Introduction and Interaction:

After accessing the meteorological department and the real-time traffic status API, we can predict the real-time probability of traffic accidents in the relevant area.

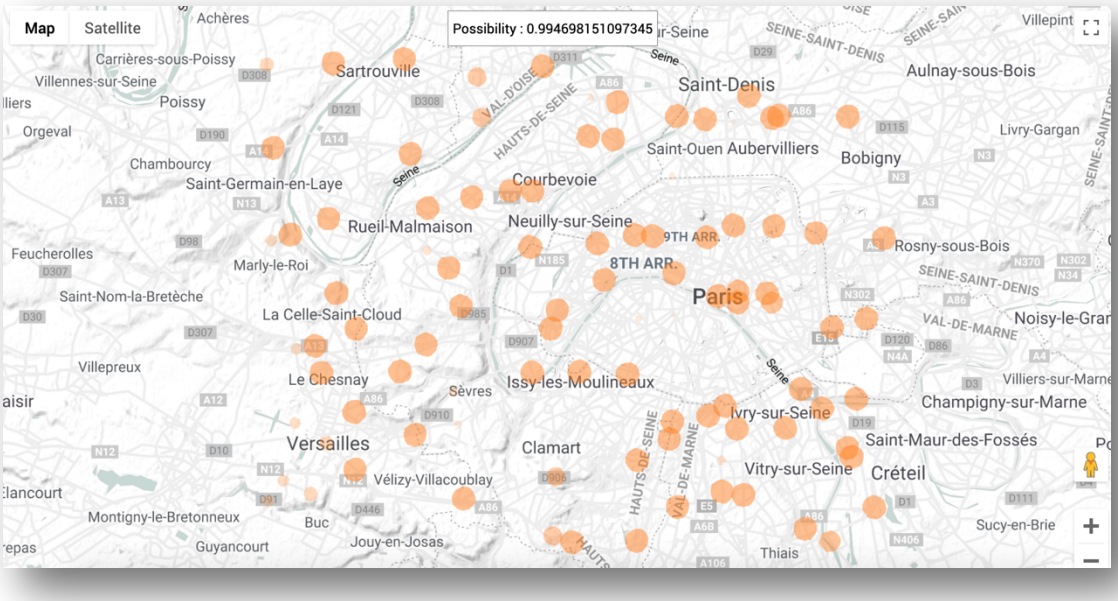
In order to show that we specially selected some scenes to simulate the access to the real-time information API.

The radius of the circle is related to the probability. The larger the probability, the larger the radius of the circle.





- After clicking on the circle, you can see the probability of a traffic accident it represents above.

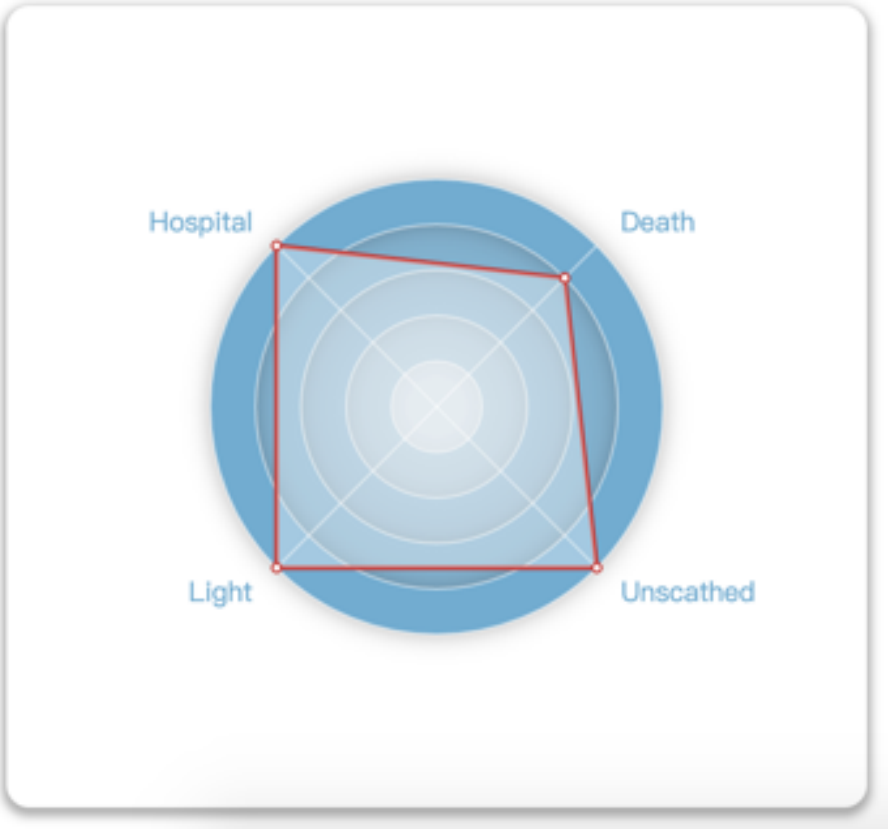


⑥ Number of people predictive results visualization with radar diagram and location map

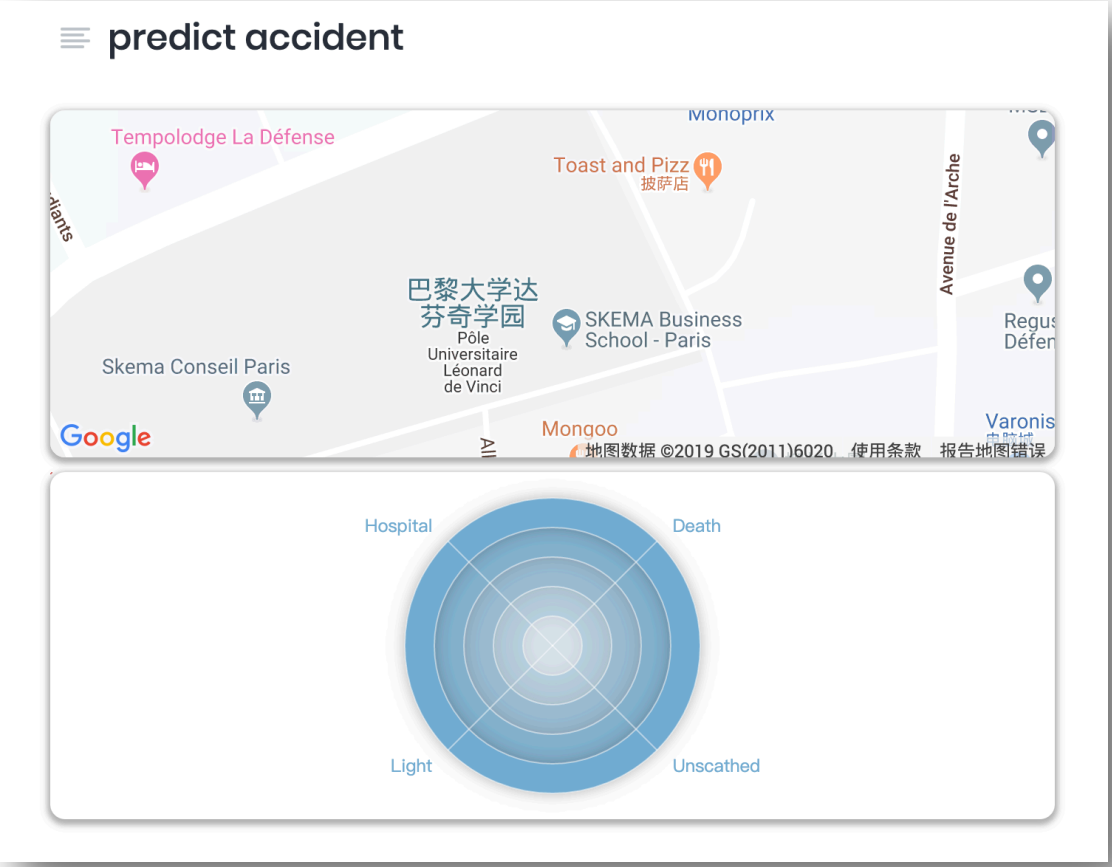
- Introduction

We display the predicted the number about "death", "hospitalization (hospital)", "minor injury (light)", and "no injury (unscathed)" of designated accident conditions

through radar chart, and display the location corresponding to latitude and longitude.



It is used in the "Predict" page in our website.

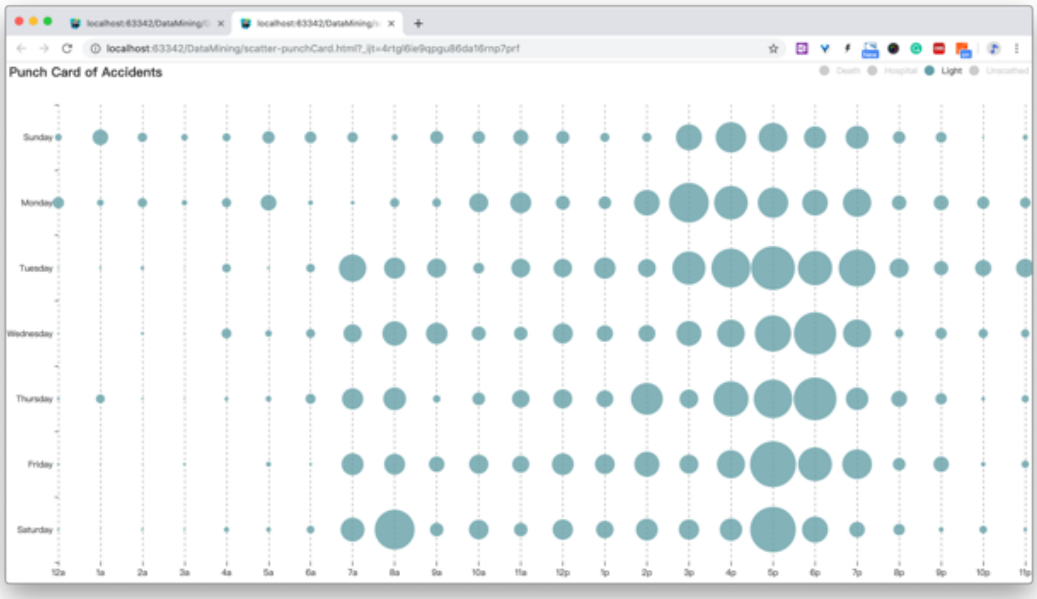


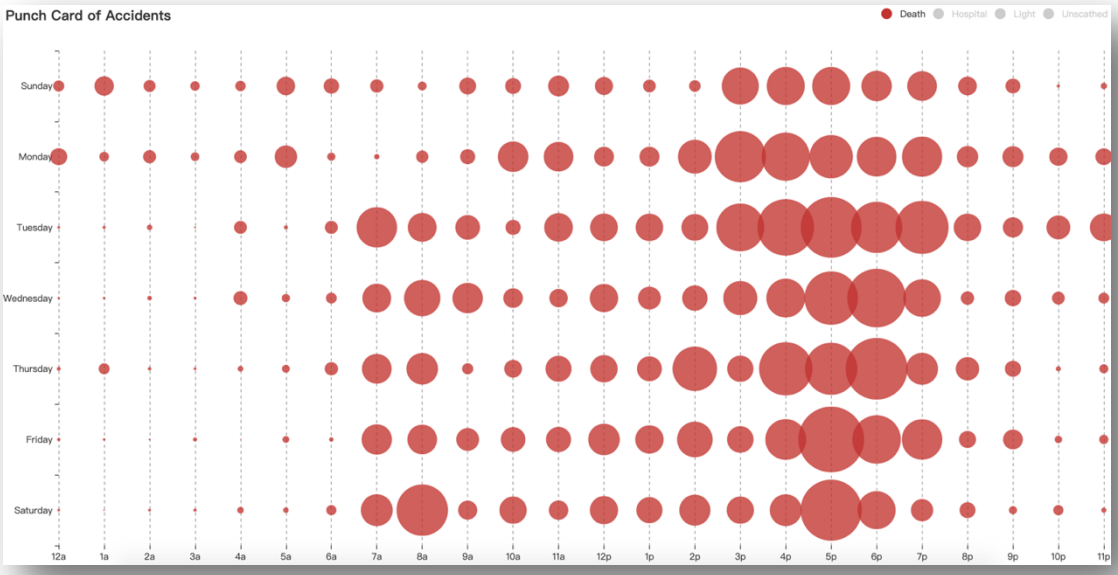
⑦ Punch card with the data of Week and 24 hours from 2005 to 2016

- Introduction:

- The abscissa is a 24-hour segment and the ordinate is a seven-week segment. The corresponding position of the horizontal and vertical coordinate intersection is one punched hole, and the radius of the hole is determined according to the number of people. The data for each punched hole is the sum of the values represented by the week and time for 2005-2016.

- In the upper right corner there is a sort selector that allows you to select the corresponding punch card.





- Place the mouse on the corresponding hole to view the specific value

- Analysis

- Through the radius of the holes, we can clearly see the severity of the accident at each time period. For example, we can see that the most serious accident often occurrs at 5-6 pm in Friday and Saturday.

6 Deployment and Testing (Step IV)

6.1 Technical Selection

6.1.1 Python 3.5.2

Used to load our machine learning model and do some logic.

6.1.2 Flask 1.0.2

Used to deploy our back end onto internet.

6.1.3 Gunicorn 19.9.0

Used to load flask project on the designated port in server.

6.1.4 Nginx 1.12.2

Used to load front end and reverse proxy the back end.

6.2 The Deployment

First, use Git to upload the project to the server.

```
#git clone https://github.com/hshsilver/TrafficAccidentsInFrance
```

Then, check the python environnement. Our project is using Python 3.5.2, but I think the other version is also available.

```
#Python -V
```

Confirm the installed package is all correct.

```
#pip install flask

#pip install gunicorn

#pip install sklearn

#pip install pandas
```

Then, cd to the back-end dirctory, use gunicorn to load flask project on the port 2500 with 4 work threads.

```
#gunicorn -w 4 -b 0.0.0.0:2500 start:app
```

Till this moment, the back end is completed. Now we need to deploy the web server and the proxy of back end by Nginx.

Our server is "CentOS Linux release 7.5.1804 (Core)" , so we use this command to install nginx.

```
#yum install nginx
```

And then, we should config the nginx to load the front end on port 80 and proxy the 'post' http method to port 2500.(Beacuse the front end http requirment all use 'post' to connect with back end.)

```
#vi /etc/nginx/nginx.conf
```

This is the config we used.

```
        server {
            listen          80;
             server_name    0.0.0.0;
            location / {
          root /root/workplace/TrafficAccidentsInFrance/Front_End;
                index index.html;
                        if ($request_method = POST) {
                    proxy_pass http://127.0.0.1:2500;
                }
            }
        }
    }
```

Finally, we just need start the server.

```
#systemctl start nginx
```

Also, we can use these cmd to control the server:

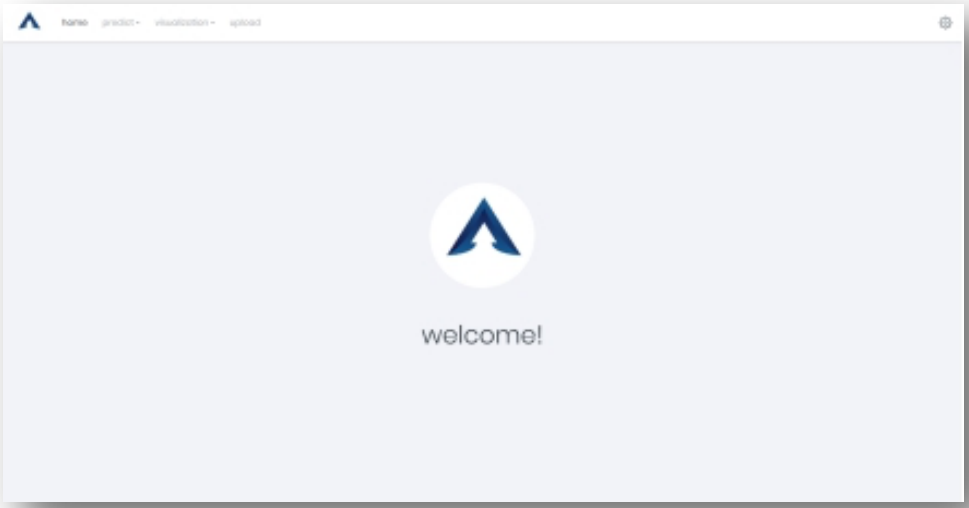
```
#systemctl start/stop/reload nginx
```

6.3 Product

You can visit our website with http://fr.hshsilver.com or http://108.160.131.50.

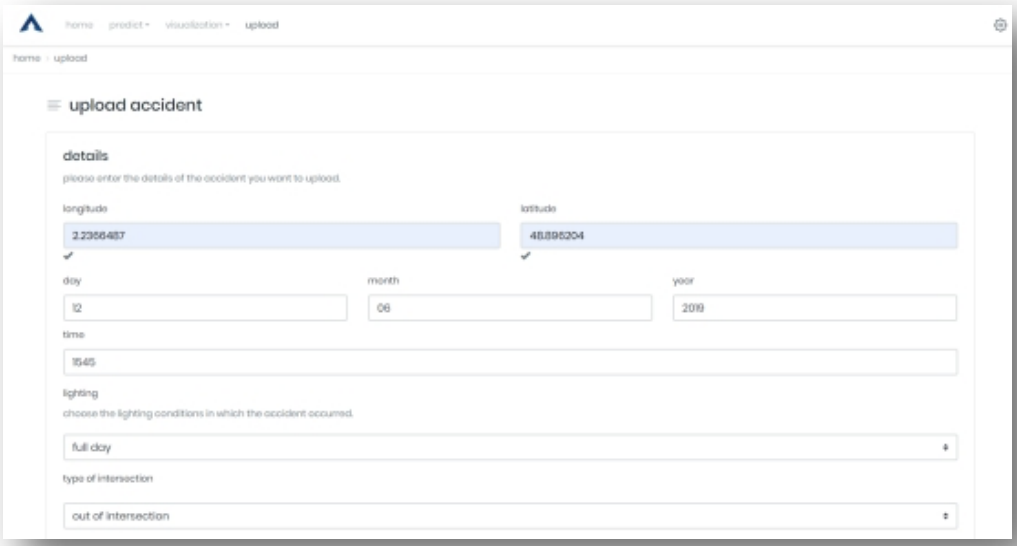
6.4. Program the application and make the main tests

6.4.1 The Home Page



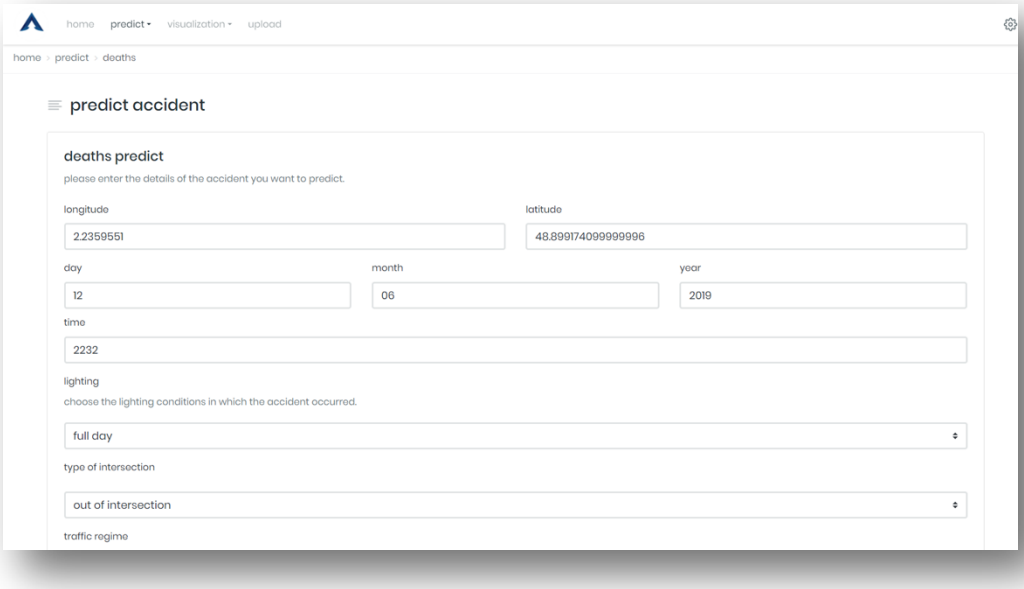
The home page is just like this, there is a top navbar to link different funtion and there is a setting button on the left top to change the language.

6.4.2 The Upload

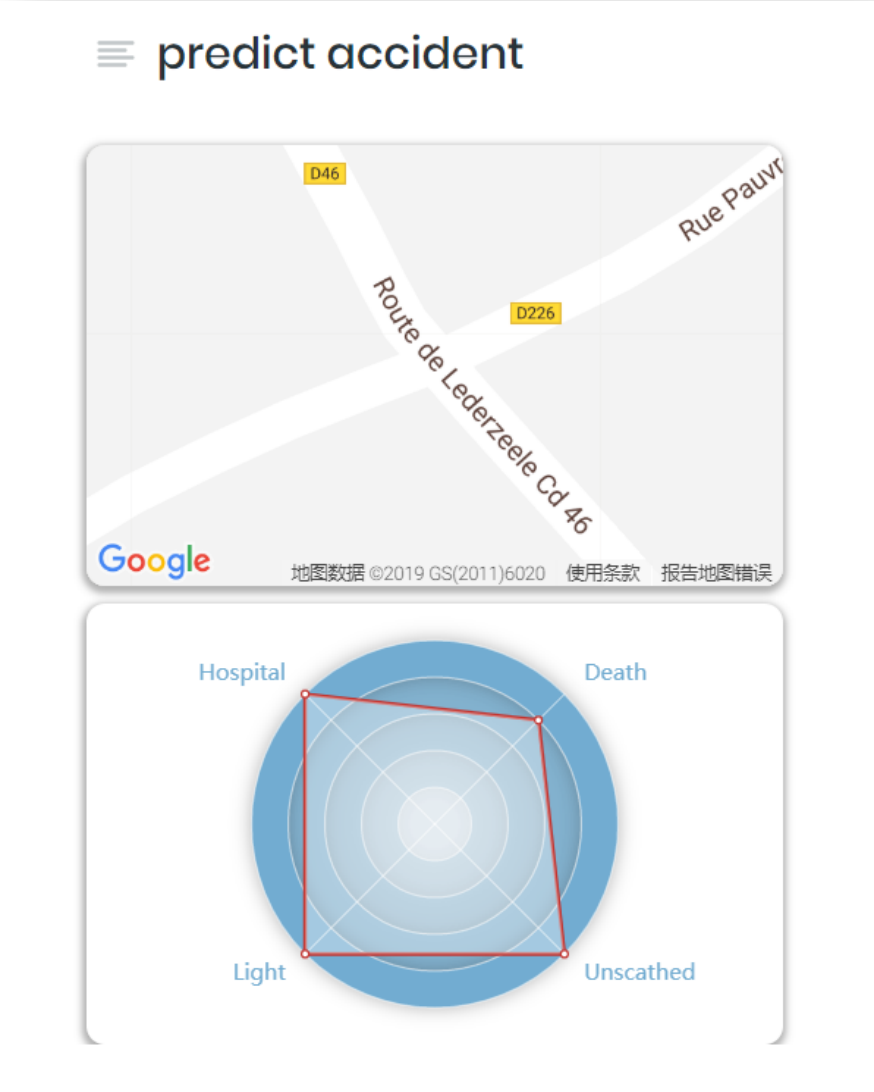


The user can upload any accident which occur around him and we will put it into visualization once we have checked it correctly.

6.4.3 The Predict of Deaths

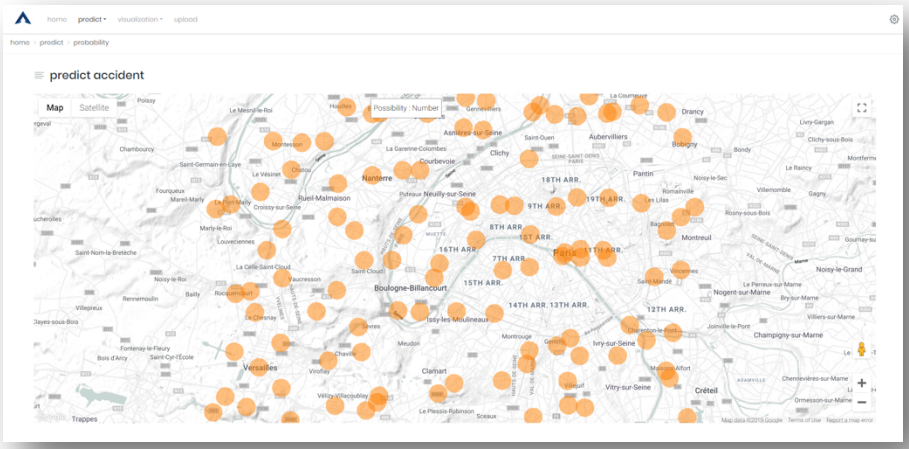


We can use deaths predict function to predict the number of deaths(include unscathed, killed, hospital and light) in an accident which user want to know.And this is the result:



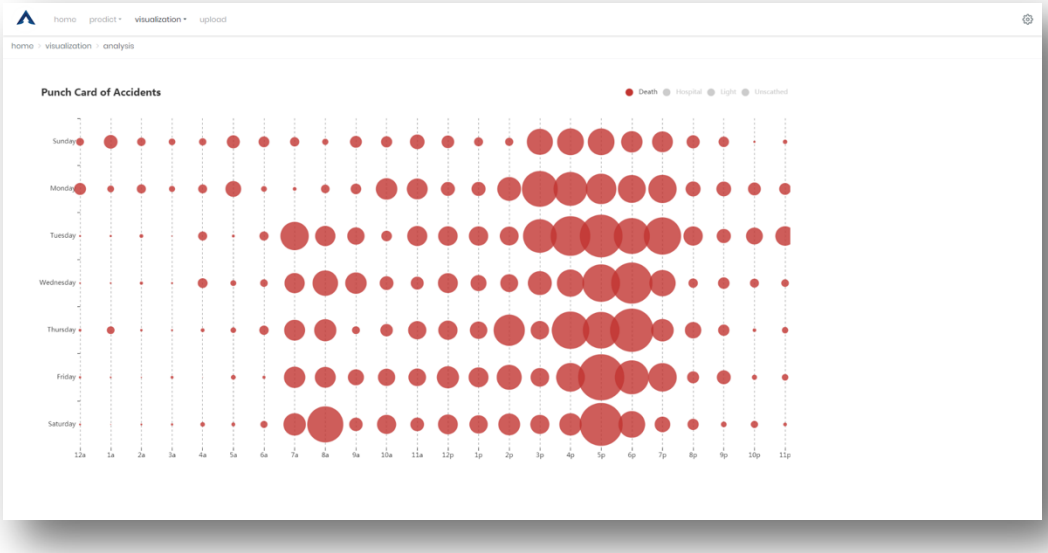
Now, we can see the predicted result of four kinds of people and also can see the really localtion on the map.

6.4.4 The Predict of Probability



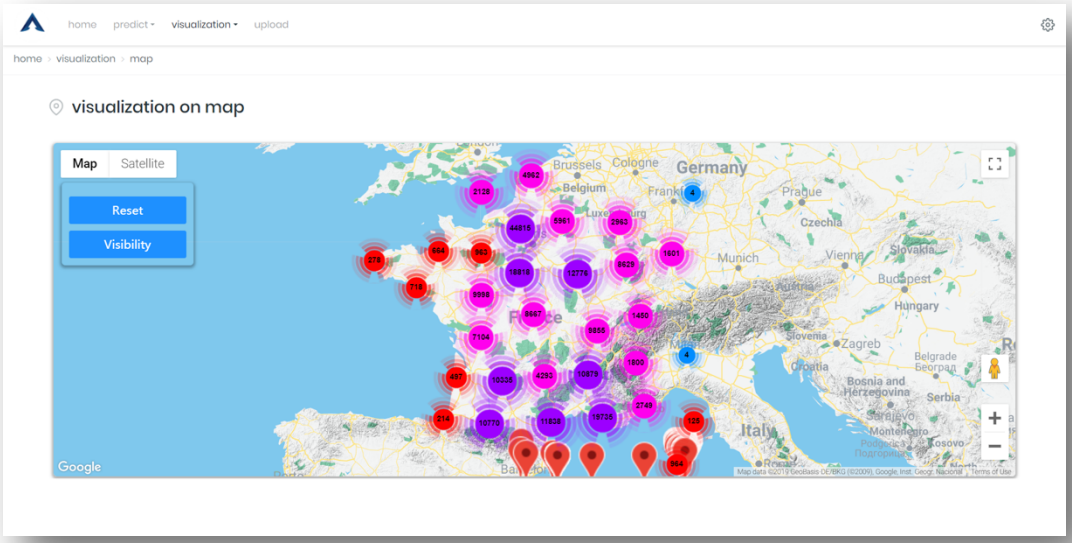
Because of missing the details of real road data, we intend to leave this part to professionals such as the state. If they upload the real information of every road, this map will show you the probability of an accident at these locations. But now, it just a mock information.

6.4.5 The Visualization on Punch Card

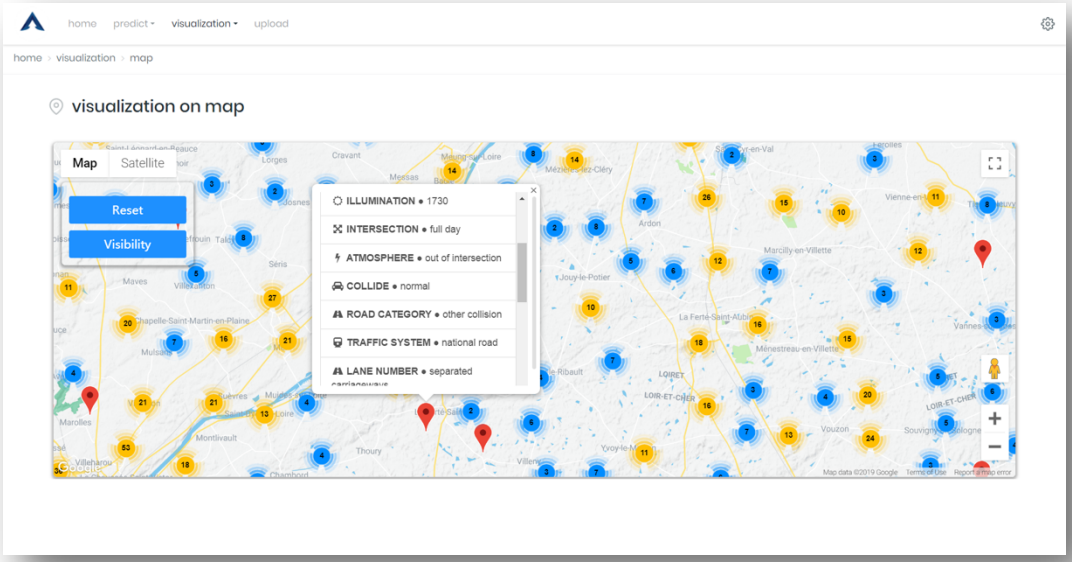


In this part, we can clearly see the number of people who have accidents every week and every hour. So it can tell us when is the most dangers time.

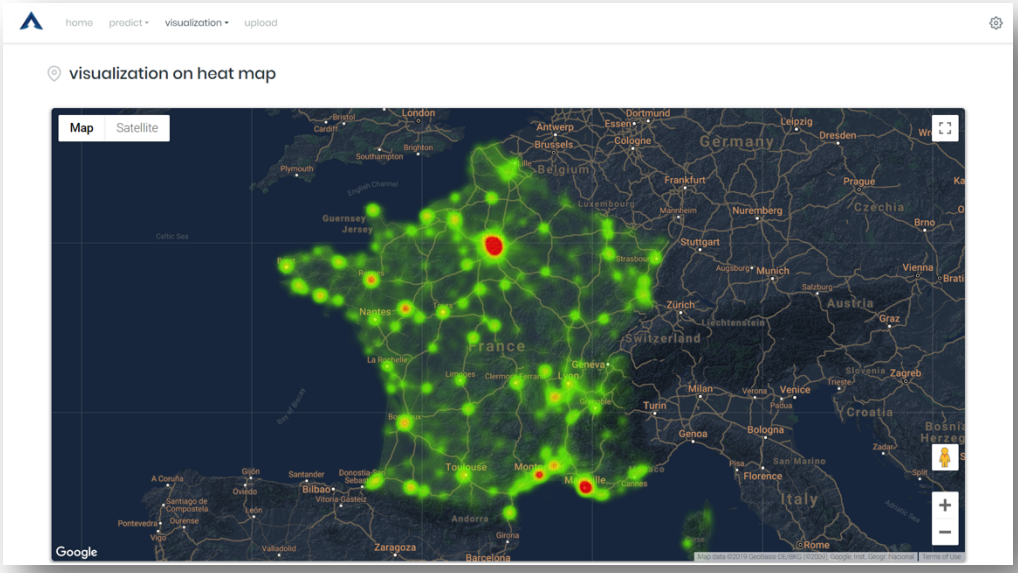
6.4.6 The Visualization on Map



We can visit all the data(200,000 items) on this map and can zoom in and out anywhere. What's more, if we click one accident, then the details of this accident will show out by a list.such as this:



6.4.7 The Visualization on Heat Map



This map clearly shows us which areas are more danger.

6.4.8 The Visualization on Pie Chart

The result of this part tell us what the proportion of people in these four kinds and we can see the number of injured people in any day in any month.

6.4.9 The Visualization on Vertical Chart

We can see the number of injured people of every year and it also shows the detail of the people in a day.

7 Conclusion

i. Advantages and Disadvantages

①Advantages:

Machine learning model can predict an accident almost accurately

The data set consists of a large amount of data and can correctly reflect the traffic accident trend in France.

②Disadvantages:

The processing of the dataset is not good enough

Understanding of the association between data items needs to be improved

i. Summary

In this project, we concentrate on data processing and machine learning. Through this 50-day study and practice, we learned about the construction of machine learning models and useful data processing knowledge. we has a better understanding about data cleaning and data reduction. Through the establishment of a machine learning model, we learned about regression and classification algorithms and evaluation methods of models.

Whet's more, it is a great chance for us to practice visualization skills, which helps us to improve both aesthetics and the ability to code.

Last but not least, we also learn how to deploy our project, borrow a domain name and combine front end and back end.

Technology has changed our life, such as the invention of cars.And technology is keeping changing our life such as the algorithm about machine learning. We do believe that after the continuous application of data mining and machine learning, we can make our lives better than better.