空难+UCL+Tweet数据集

zhdzu

一、1908-2009空难数据集

1.1 主要工具：matplotlib\seaborn\pyecharts\nltk pyecharts\nltk为主
1.2 空难次数随年份、月份和小时变换情况
1.3 空难次数随年份和月份交替的 3D 图
1.4 地图可视化
    1.4.1 pyecharts编辑美国各州信息，全球地区信息
    1.4.2 数据中地理信息和pyecharts中地区一一对应
    1.4.3 地区词云图
    1.4.4 美国各州发生空难可视化
    1.4.5 全球各地区发生空难可视化
    1.4.6 排名靠前的城市可视化
1.5 伤亡分析
    1.5.1 地面伤亡人数分析
    1.5.2 每年登机和遇难人数分析
1.6 机型分析

1.7 Summary

1.7.1 分词处理
1.7.2 小写处理
1.7.3 除标点符号和停用词
1.7.4 词干化处理
1.7.5 简单的统计汇总
1.7.6 上下文相关内容
1.7.7 画词云图

1.8 遇难原因探究

1.9 词性标注

Content

Data format: Format

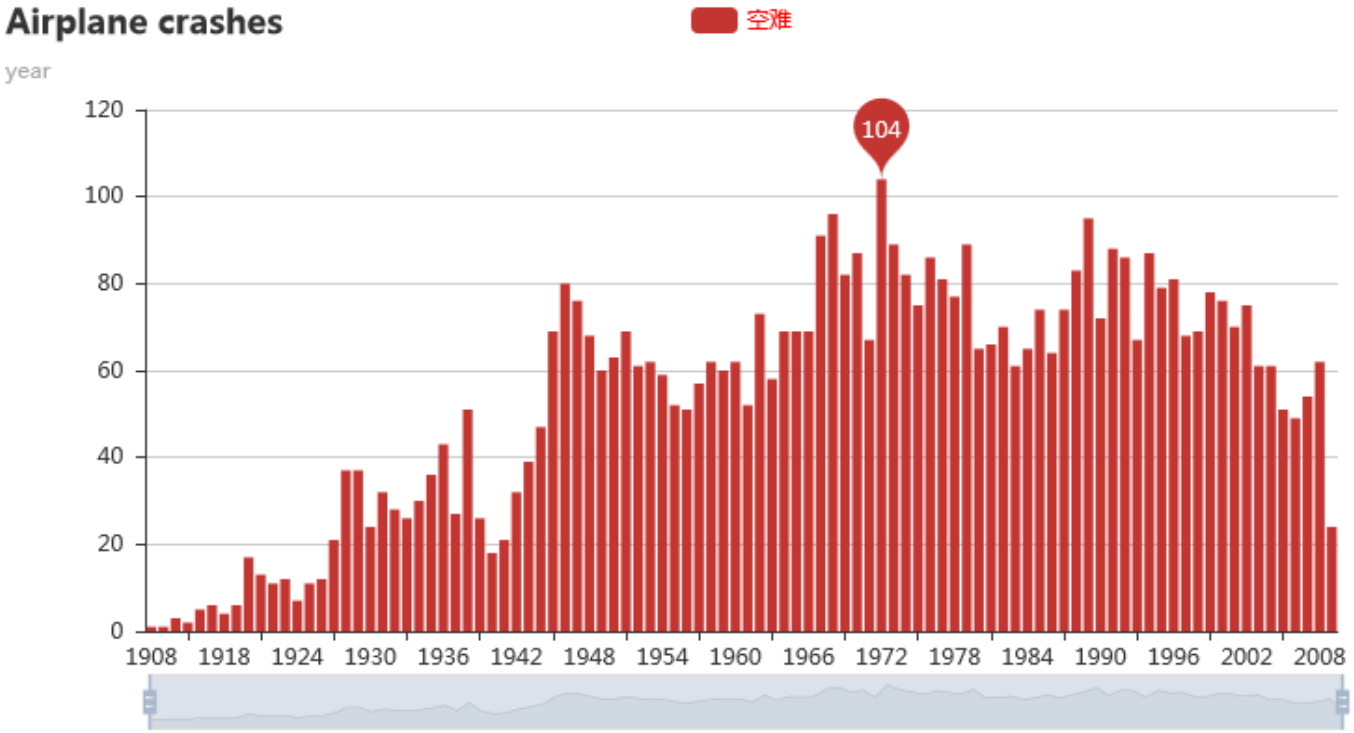
```
date:    Date of accident,  in the format - January 01, 2001
time:    Local time, in 24 hr. format unless otherwise specified
location: location information
Airline/Op:  Airline or operator of the aircraft
flight_no:   Flight number assigned by the aircraft operator
route:   Complete or partial route flown prior to the accident
ac_type:     Aircraft type
registration:    ICAO registration of the aircraft
cn_ln:   Construction or serial number / Line or fuselage number
aboard:  Total aboard (passengers / crew)
fatalities:  Total fatalities aboard (passengers / crew)
ground:  Total killed on the ground
summary:     Brief description of the accident and cause if known
```

https://www.kaggle.com/nguyenhoc/plane-crash/home

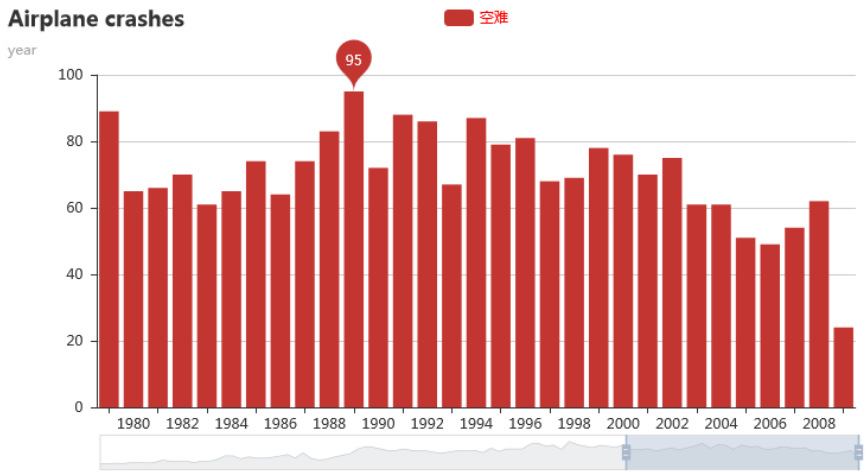
本文选取
时间变量：date、time
地点变量：location
分类变量：ac_type
连续型变量：aboard、
fatalities、ground
本文变量：summary
进行分析。

credit.head( ) credit.info( ) credit.describe( ) 函数进行数据初探，此处不列举。

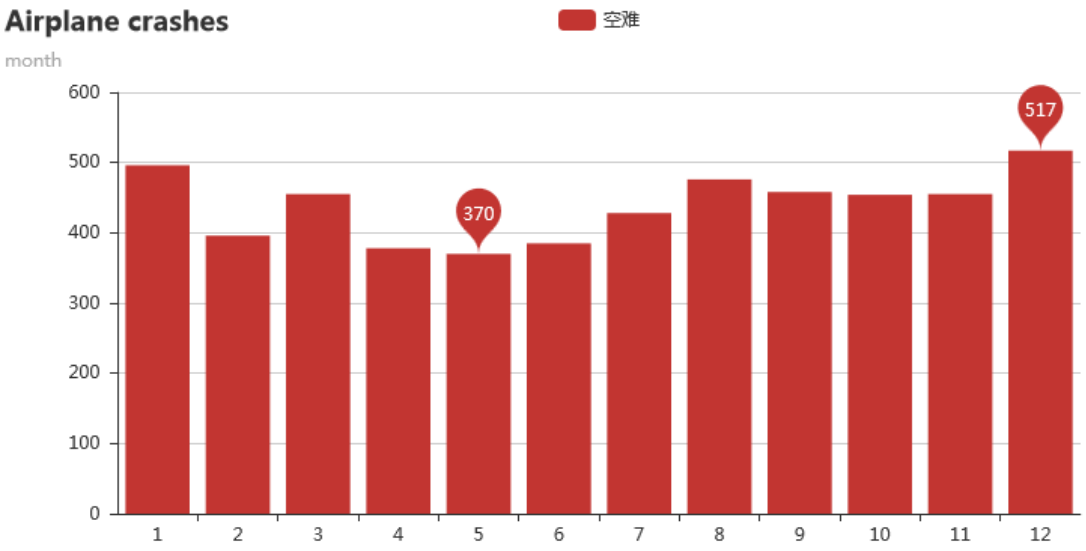
空难次数随年份变化情况



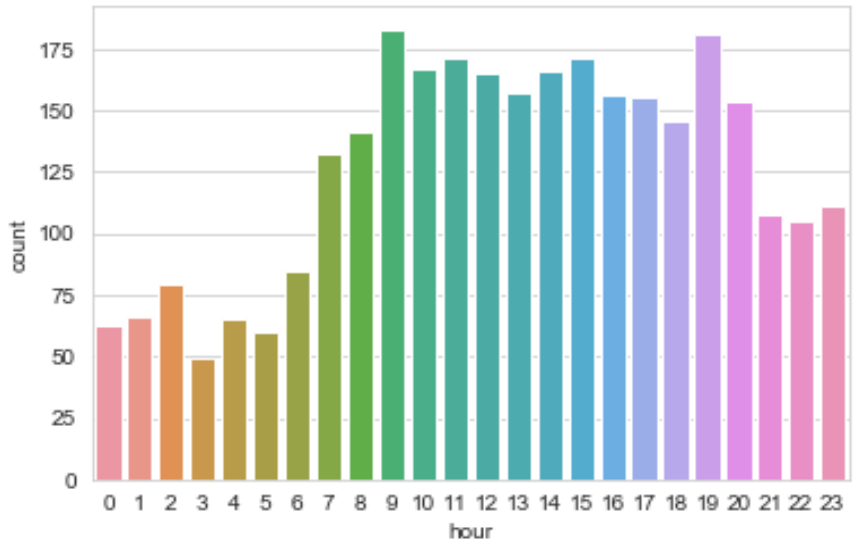
1972年发生空难次数最多，达到了104次。



空难次数随月份和小时变化情况



4-6月发生空难次数较少，12-1月发生空难次数较多。



一天中早上9点到晚上8点发生空难次数较多。

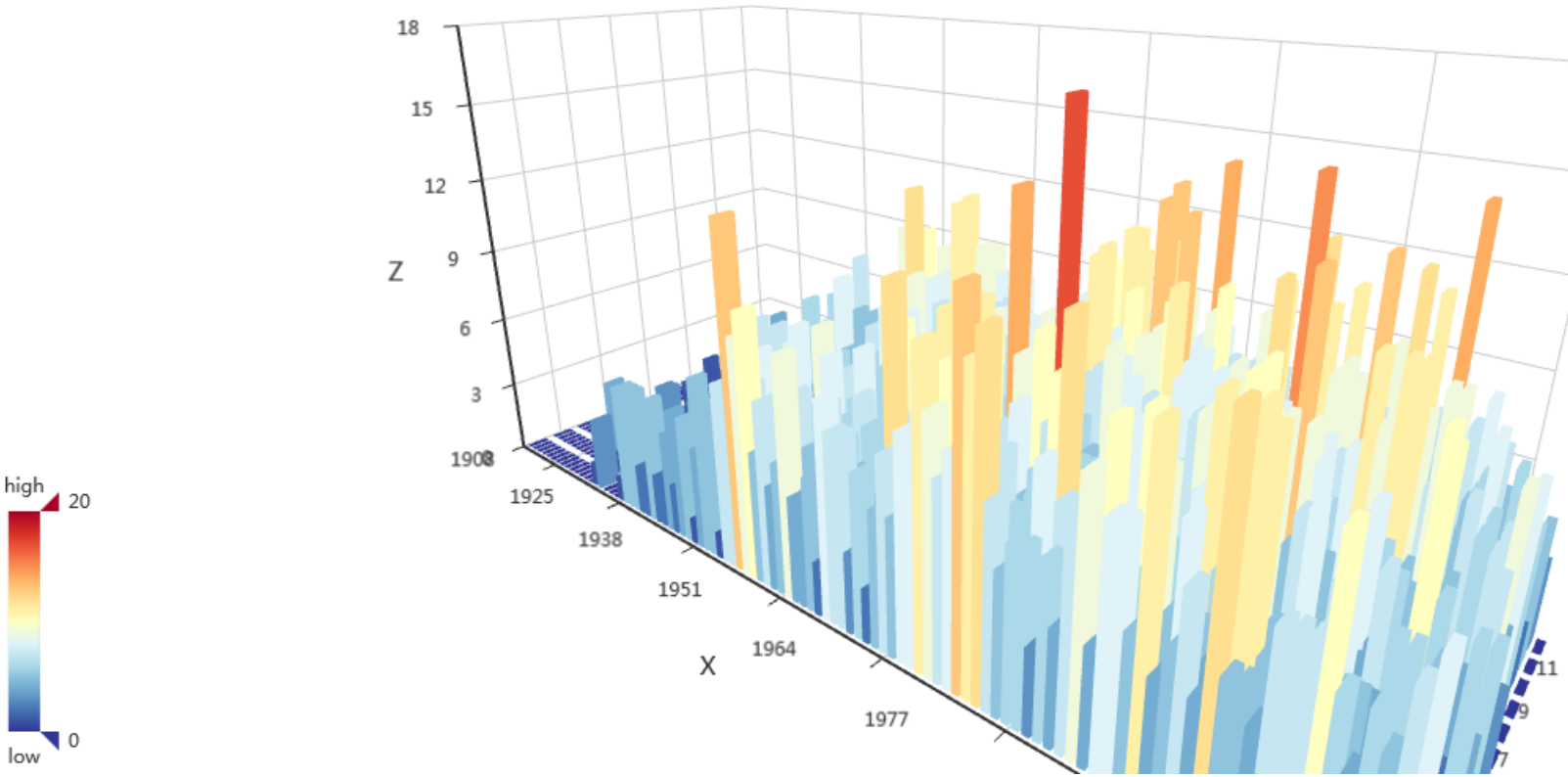
年份和月份交替的 3D 图

```
# 构建三维数据集
# (y,x,z)格式
data1 = []
for i in range(102):
    for j in range(12):
        data1.append([j,i,data[i,j]])
data1[0:6]
```

```
[[0, 0, 0.0], [1, 0, 0.0], [2, 0, 0.0], [3, 0, 0.0], [4, 0, 0.0], [5, 0, 0.0]]
```

空难分布 3D 柱状图

year-month



空难发生地点可视化

地址总数: 4570
词汇丰富度: 2.3308533916849017
查找地名出现次数多于15次的个数: 501
查找地名出现次数多于10次的个数: 2306

```
# 出现最频繁的地点
fd =nltk.FreqDist(words).most_common()
fd
```

```
[('Brazil', 189),
 ('Alaska', 177),
 ('Russia', 175),
 ('Canada', 149),
 ('Colombia', 149),
 ('California', 143),
 ('France', 133),
 ('England', 105),
```

数据集和pyecharts中美国州名一一对应

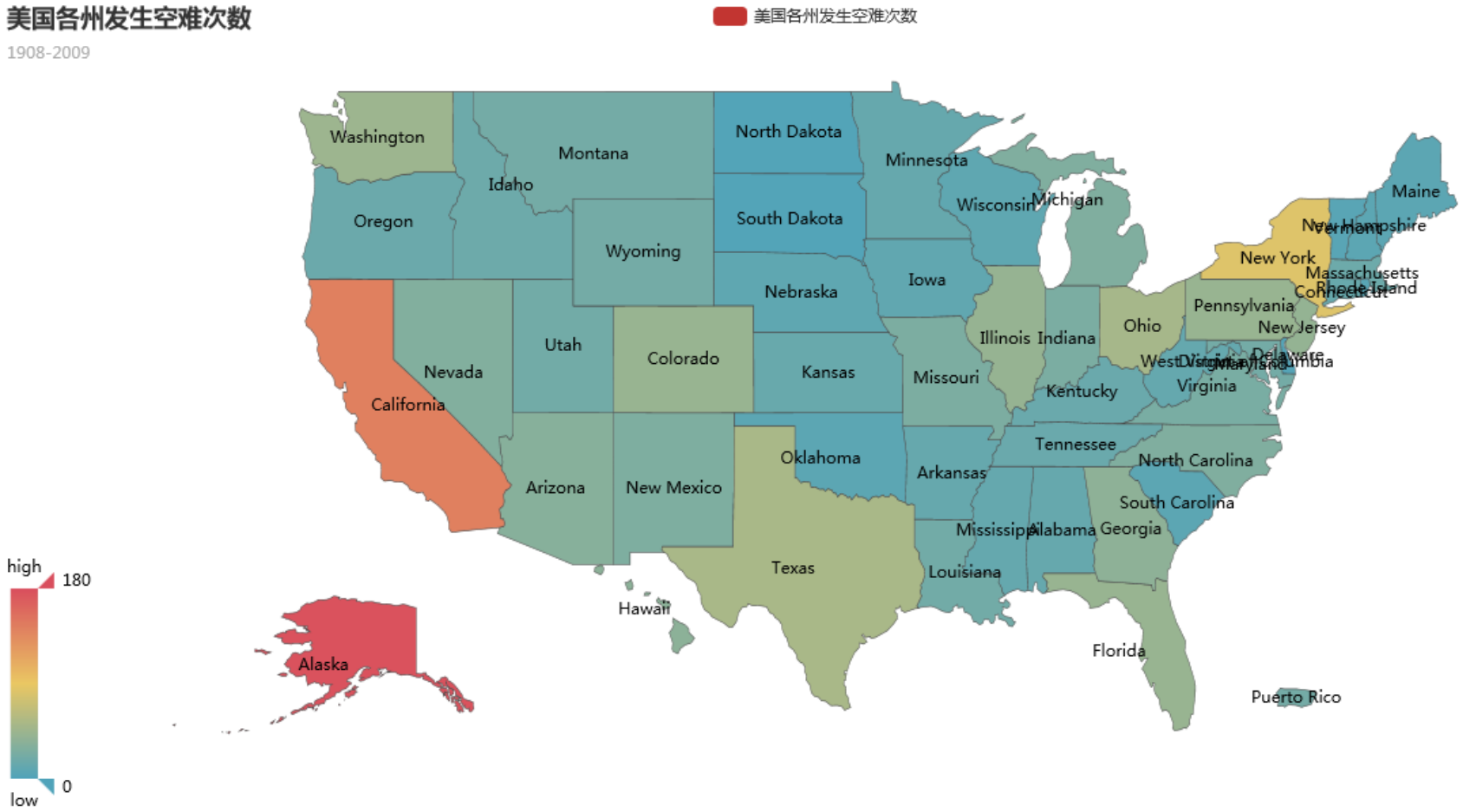
```python
# 统计各州发生空难的人数
states_usa_words = [
        'Alabama','Alaska','Arizona','Arkansas','California',
        'Colorado','Columbia','Connecticut','Delaware','Florida',
        'Georgia','Hawaii','Idaho','Illinois','Indiana',
        'Iowa','Kansas','Kentucky','Louisiana','Maine',
        'Maryland','Massachusetts','Michigan','Minnesota','Mississippi'
        'Missouri','Montana','Nebraska','Nevada','NewHampshire',
        'NewJersey','NewMexico','NewYork','NorthCarolina','NorthDakota'
        'Ohio','Oklahoma','Oregon','Pennsylvania','RhodeIsland',
        'SouthCarolina','SouthDakota','Tennessee','Texas','Utah',
        'Vermont','Virginia','Washington','WestVirginia','Wisconsin',
        'Wyoming','PuertoRico']

states_usa_num = [words.count(state) for state in states_usa_words]
```

```python
states_usa_pyecharts = [
        'Alabama','Alaska','Arizona','Arkansas','California',
        'Colorado','Columbia','Connecticut','Delaware','Florida',
        'Georgia','Hawaii','Idaho','Illinois','Indiana',
        'Iowa','Kansas','Kentucky','Louisiana','Maine',
        'Maryland','Massachusetts','Michigan','Minnesota','Mississippi',
        'Missouri','Montana','Nebraska','Nevada','New Hampshire',
        'New Jersey','New Mexico','New York','North Carolina','North Dakota',
        'Ohio','Oklahoma','Oregon','Pennsylvania','Rhode Island',
        'South Carolina','South Dakota','Tennessee','Texas','Utah',
        'Vermont','Virginia','Washington','West Virginia','Wisconsin',
        'Wyoming','Puerto Rico']
```

```python
# 需要将pyecharts和words中的洲名一一对应
# New Hampshire -- NewHampshire     # New Jersey -- NewJersey     # New Mexico -- NewMexico
# North Carolina -- NorthCarolina   # New York -- NewYork        # North Dakota -- NorthDakota
# Rhode Island -- RhodeIsland       # South Carolina -- SouthCarolina
# South Dakota -- SouthDakota       # West Virginia -- WestVirginia
# Puerto Rico -- PuertoRico
```

Text1：数据集中地名；
Text2：美国州名
从Text2中复制州名去Text1中进行查找，从而对所有州一一对应，比较费时。

美国各州发生空难次数


美国各州发生空难次数
1908-2009
美国各州发生空难次数
Washington
Montana
North Dakota
Minnesota
Idaho
Wisconsin
Michigan
Maine
Oregon
South Dakota
Vermont
New Hampshire
Wyoming
New York
Massachusetts
Iowa
Nebraska
Rhode Island
Connecticut
Pennsylvania
Ohio
New Jersey
Illinois
Indiana
Utah
Colorado
Kansas
Missouri
Nevada
Kentucky
Virginia
California
Tennessee
North Carolina
Oklahoma
Arkansas
Arizona
New Mexico
South Carolina
Mississippi
Alabama
Georgia
Texas
Louisiana
Hawaii
Florida
Alaska
Puerto Rico
high
180
0
low

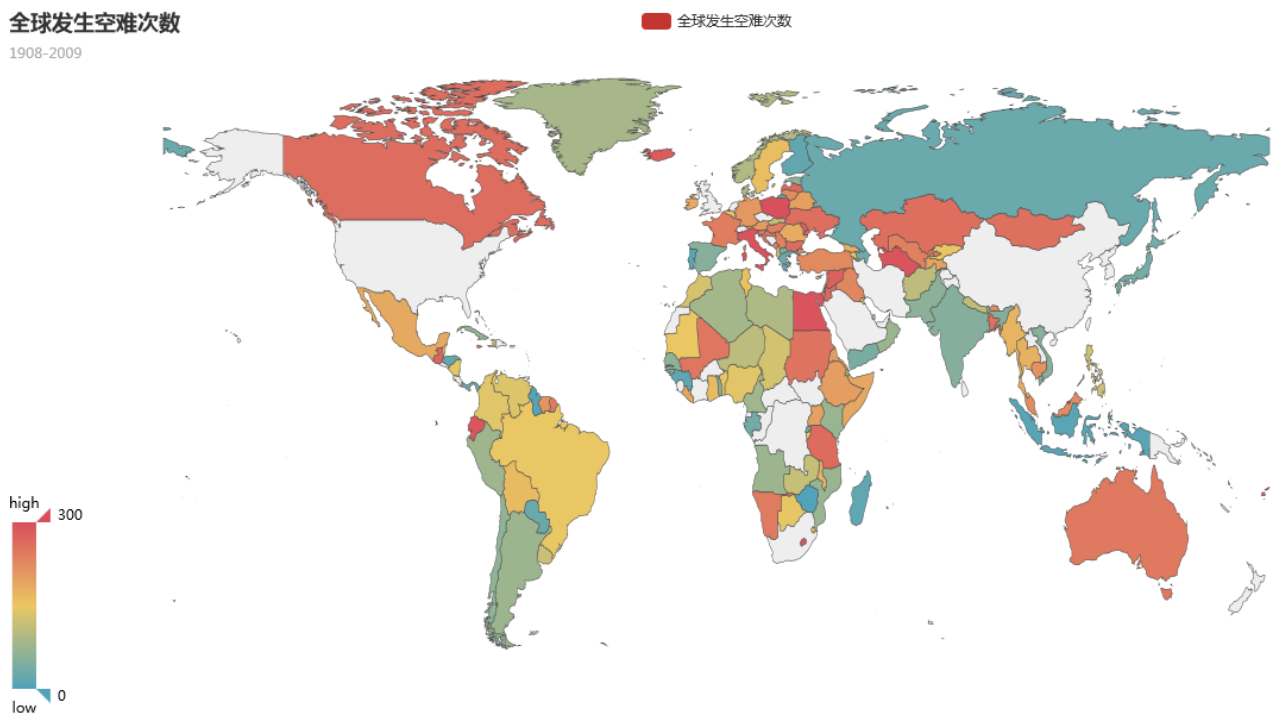
最费时间的事——1. 整理出pyecharts中所有国家的地名

Featuring Cities(or for Single Download)

Cities:

1. 阿富汗 (Afghanistan)
2. 阿尔巴尼亚 (Albania)
3. 阿尔及利亚 (Algeria)
4. 安道尔 (Andorra)
5. 安哥拉 (Angola)
6. 安圭拉 (Anguilla)
7. 安提瓜和巴布达 (Antigua and Barbuda)
8. 阿根廷 (Argentina)

https://github.com/echarts-maps/echarts-countries-js



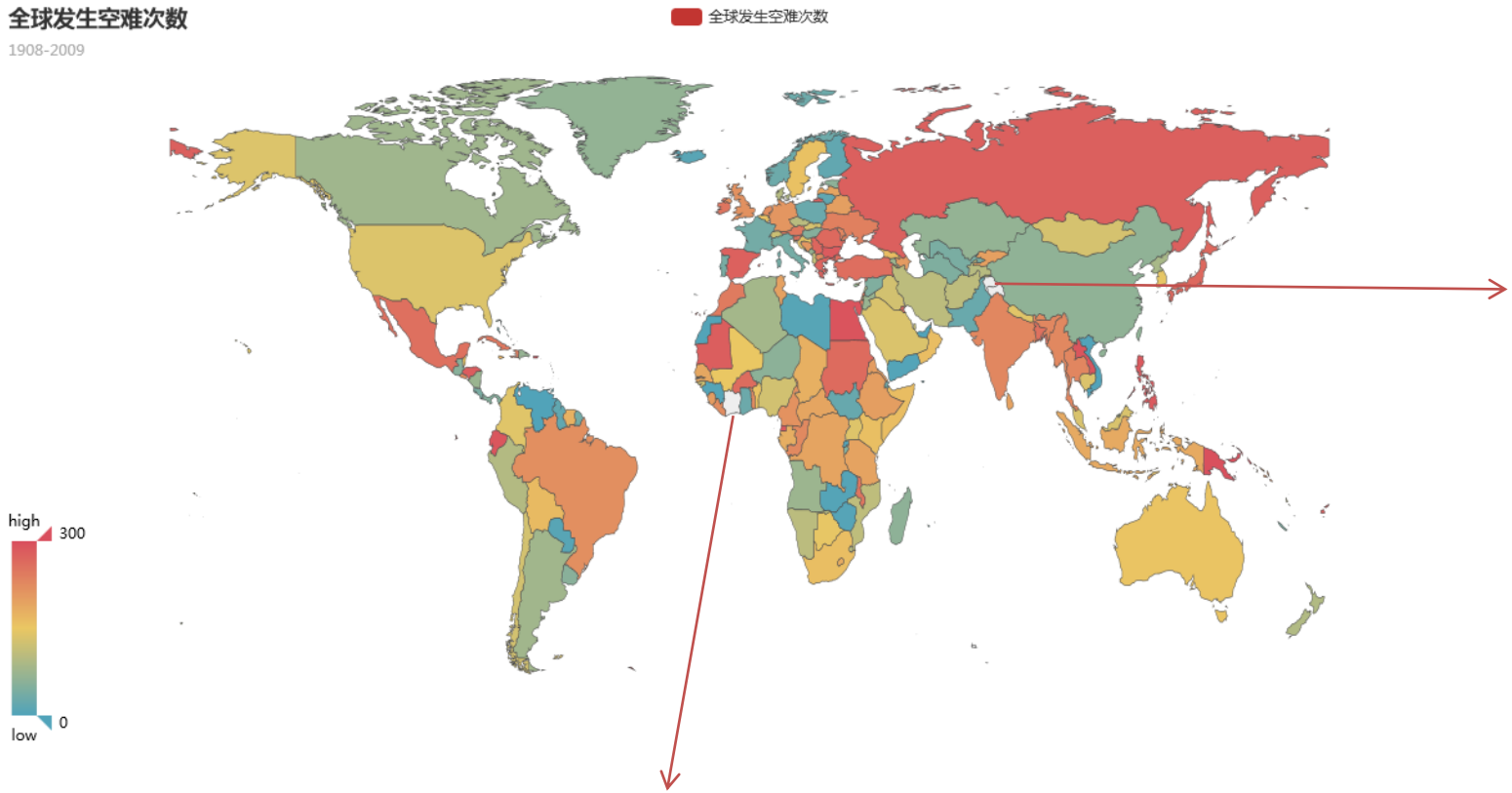
官网列出212个+china，其中部分处理结果如下（共处理35个）：

美国 USA —— United States
伊朗 Islamic Republic of Iran —— Iran
库克群岛 Cook Islands —— Cook Is.
刚果金 Congo-Kinshasa —— Congo
刚果民主共和国 Congo-Brazzaville —— Dem. Rep. Congo

最费时间的事——1 整理出pyecharts中所有国家的地名

最后整理出218个国家和地区，可能有重的。

其中还少两个地区，一个是非洲，是法语名，一个是亚洲，图上没显示名称。



最费时间的事——2. 数据集和pyecharts中全球地区名一一对应

```
print('数据集和pyecharts有多少个地名相同'+' '+str(len(same)))
print('数据集和pyecharts有多少个地名不同（去除美国51个洲）'+' '+str(len(none)))
none[0:20]
```

```
数据集和pyecharts有多少个地名相同 144
数据集和pyecharts有多少个地名不同（去除美国51个洲） 4375
```

本文对于数据集和pyecharts中地名不符的地名进行词频统计。主要处理了出现次数大于10次的地名，共36个，发现一些有意思的现象。

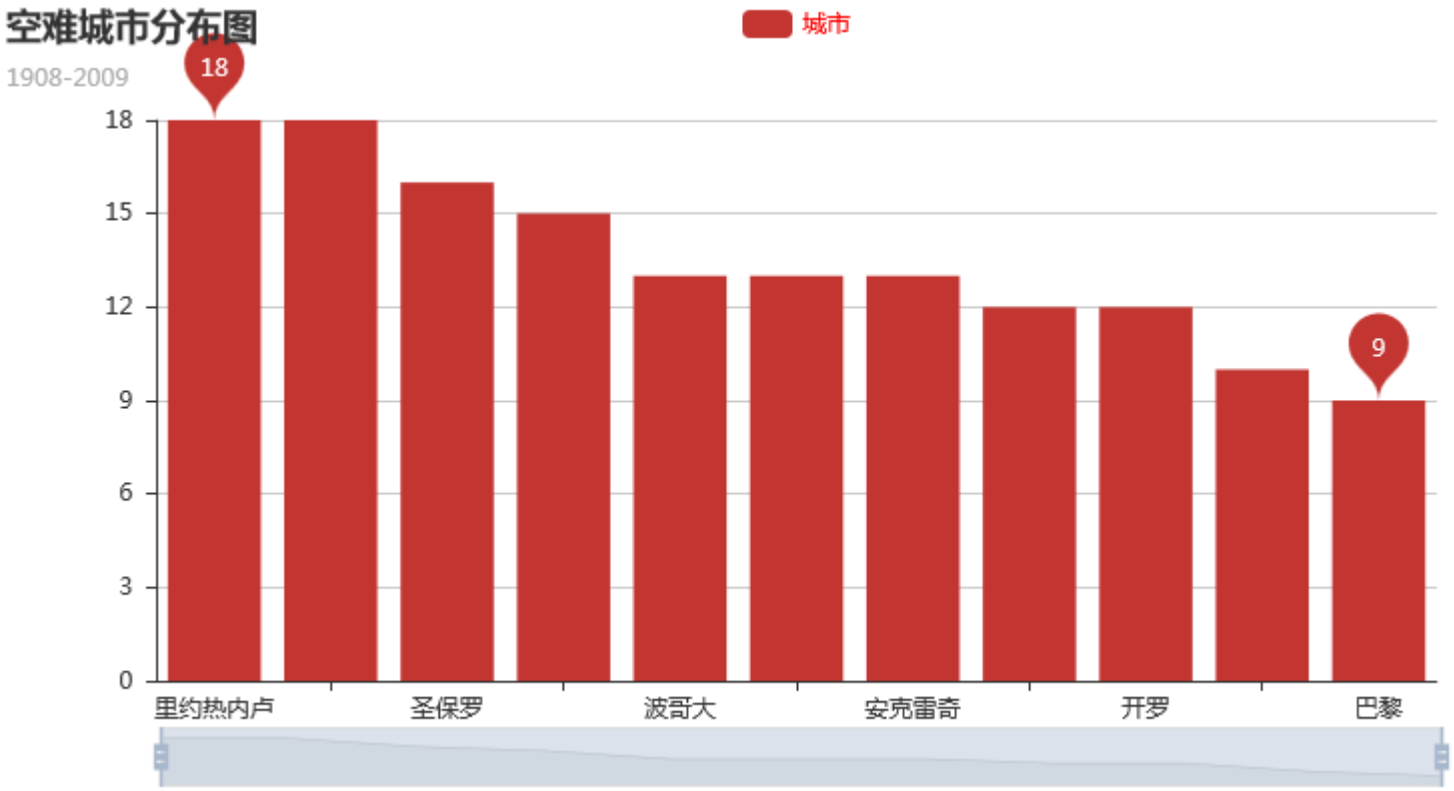
有一些二战时期的国家，有一些是城市名，有Taiwan和HongKong在数据集中没有列入中国。

加拿大：

不列颠哥伦比亚省 23
安大略省 17
纽芬兰省 17
魁北克省 14

```
'''地点出现次数大于10的均转化成pyecharts中的地名'''
# England -- 105 -- United Kingdom
# USSR -- 苏联61 -- Russia
# SouthVietnam -- 二战时的越南38 -- Vietnam
# Taiwan -- 台湾 -- China
# Laos -- 老挝26 -- Lao PDR
# NewZealand -- 纽西兰 -- New Zealand
# NewGuinea -- 巴布亚新几内亚独立国 -- Papua New Guinea
# PuertoRico -- Puerto Rico
# SouthAfrica -- South Africa
# PapuaNewGuinea -- Papua New Guinea
# Scotland -- United Kingdom
# Congo -- 刚果 -- Congo
# SouthKorea -- 朝鲜 -- Korea
# DemocratiRepubliCongo -- 刚果民主共和国 -- Dem. Rep. Congo
# SaudiArabia -- 沙特 -- Saudi Arabia
# Czechoslovakia -- 捷克斯洛伐克二战 -- Slovakia
# Czechoslovakia -- 捷克斯洛伐克二战 -- Czech Rep.
# Yugoslavia -- 二战南斯拉夫
# SriLanka -- 斯里兰卡 -- Sri Lanka
# CostaRica -- 哥斯达黎加 -- Costa Rica
# Burma -- 二战缅甸 -- Myanmar
# HongKong -- 香港 -- China
```

空难城市分布图



里约热内卢 18
大西洋 18
圣保罗 16
莫斯科 15
波哥大 哥伦比亚首都 13
马尼拉 菲律宾 13
安克雷奇市 美国 13
芝加哥 12
开罗 12
伦敦 10
巴黎 9

最费时间的事——2. 数据集和pyecharts中全球地区名一一对应

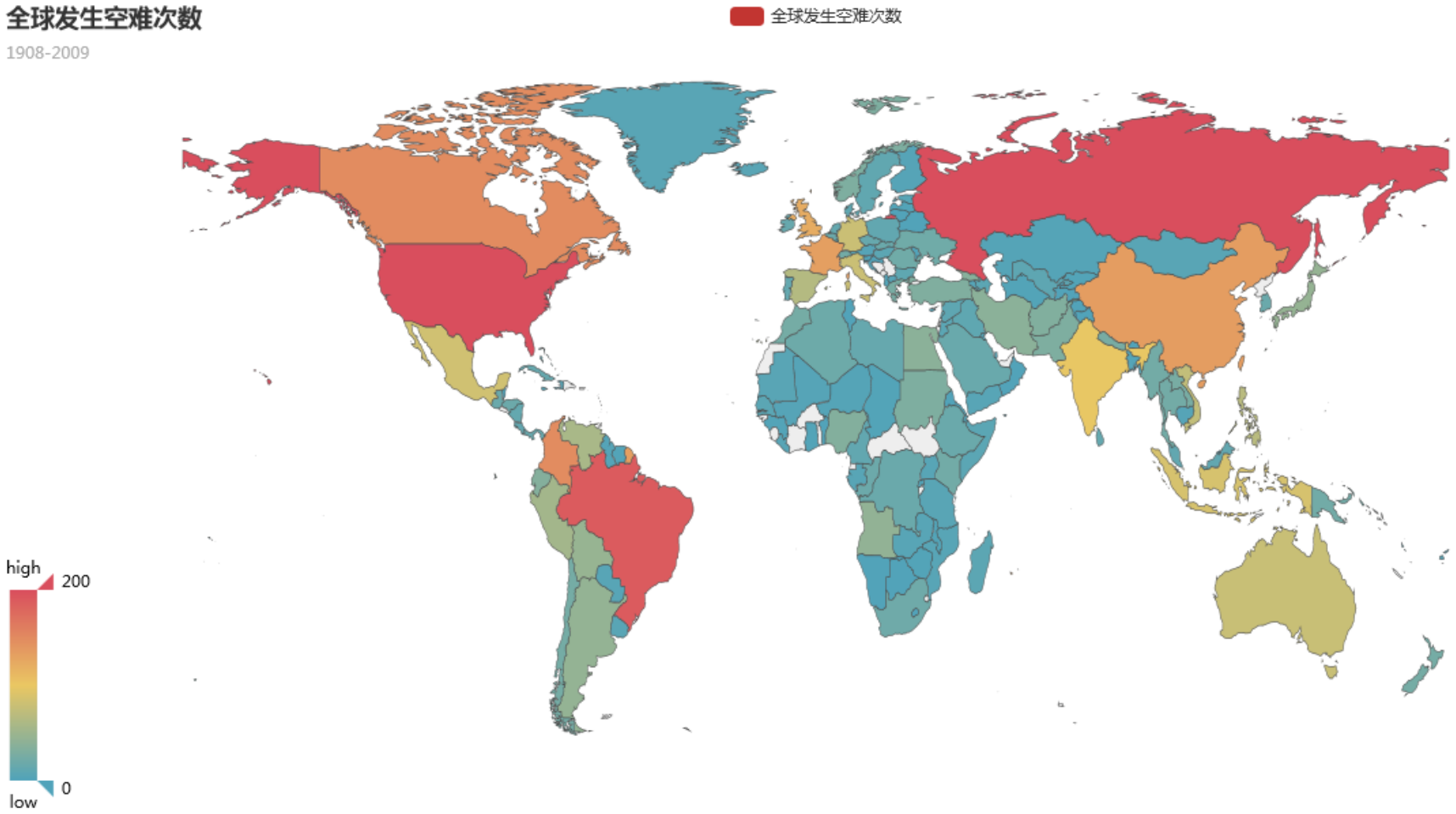
处理完出现前10次的地区后还有问题，有些国家在地图上没有出现，次数太少没有进行名称的对应，我们对这些国家进行了处理。共10个地区。

```
'SierraLeone':'Sierra Leone','EquatorialGuinea':'Eq. Guinea','BosniaHerzegovina':'Bosnia and Herz.',
'SolomonIslands':'Solomon Is.','EastTimor':'Timor-Leste','NorthKorea':'Dem. Rep. Korea',
'UnitedArabEmirates':'United Arab Emirates','CentralAfricanRepublic':'Central African Rep.',
'DominicanRepublic':'Dominican Rep.','ElSalvador':'El Salvador'}
```

```
'SierraLeone','EquatorialGuinea','BosniaHerzegovina','SolomonIslands','EastTimor',
'NorthKorea','UnitedArabEmirates','CentralAfricanRepublic','DominicanRepublic','ElSalvador']
```

其余4329个地名不相符的信息，可能为州名，城市名等，因为20-36大部分均为城市，且其出现次数小于9次，便不再处理。

全球地区发生空难次数



空难数最大值 189
空难数最小值 1
美国各州加和 1418
美国图中记作200

伤亡分析

Content

Data format: Format

```
date:    Date of accident,  in the format - January 01, 2001
time:    Local time, in 24 hr. format unless otherwise specified
location: location information
Airline/Op:  Airline or operator of the aircraft
flight_no:   Flight number assigned by the aircraft operator
route:   Complete or partial route flown prior to the accident
ac_type:     Aircraft type
registration:    ICAO registration of the aircraft
cn_ln:   Construction or serial number / Line or fuselage number
aboard:  Total aboard (passengers / crew)
fatalities:  Total fatalities aboard (passengers / crew)
ground:  Total killed on the ground
summary:     Brief description of the accident and cause if known
```

https://www.kaggle.com/nguyenhoc/plane-crash/home

变量Groud、Aboard、Fatalities
Aboard：登机人数
Fatalities：飞机上人员伤亡人数
Ground：地面人员伤亡人数

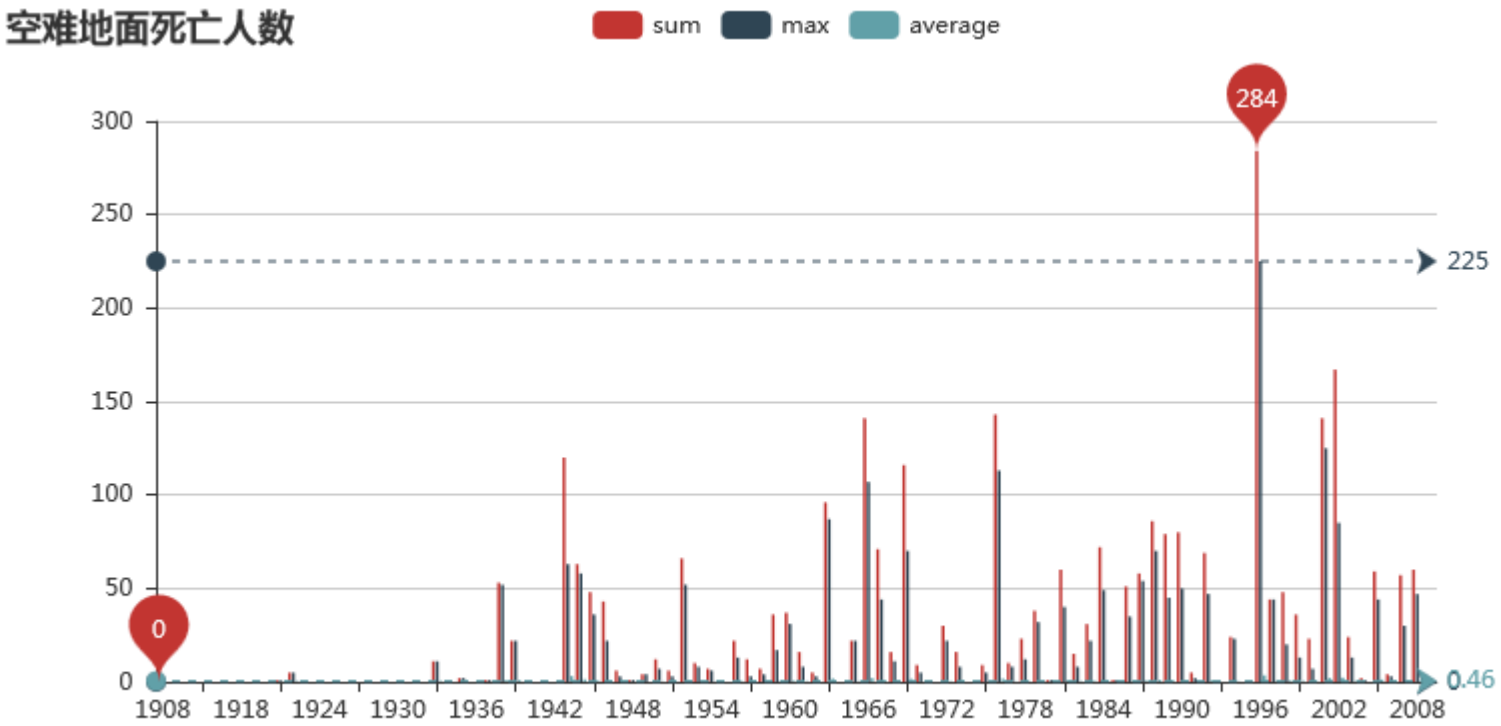
Fatalities

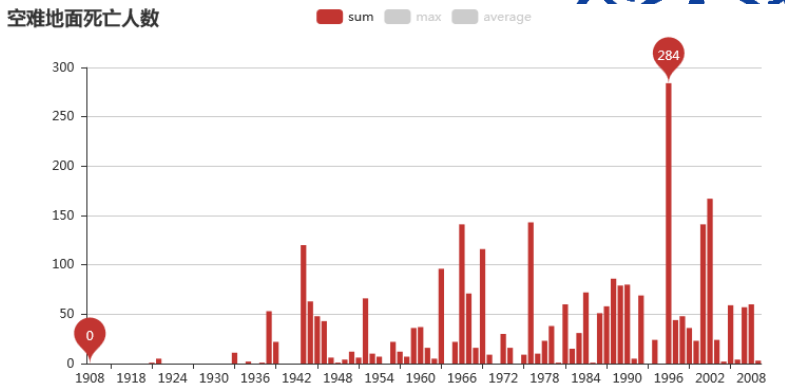
1908-2009年空难伤亡总人数 ：105479
1908-2009年空难登机总人数 ：144551
伤亡概率 ：72.97%
内特里费空难：两架波音-747跑道相撞，死亡583人，又称世纪大空难，浓雾
日航123空难：波音747撞富士山，死亡520人，单架飞机死亡人数最多，少一排铆钉，下雨
恰尔基达德里撞机事件，最严重的的空中撞机事件，死亡349人，英语口音交流不畅
土耳其航空981号班机空难：货舱门未锁定导致爆炸性施压，死亡346人
印度航空182号班机：恐怖袭击，死亡329人，炸弹爆炸

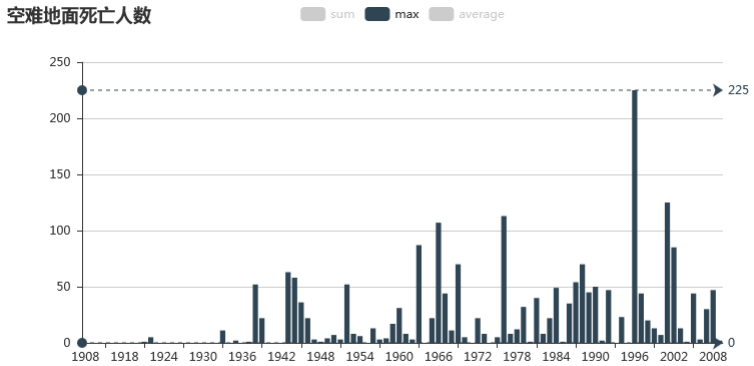
Ground

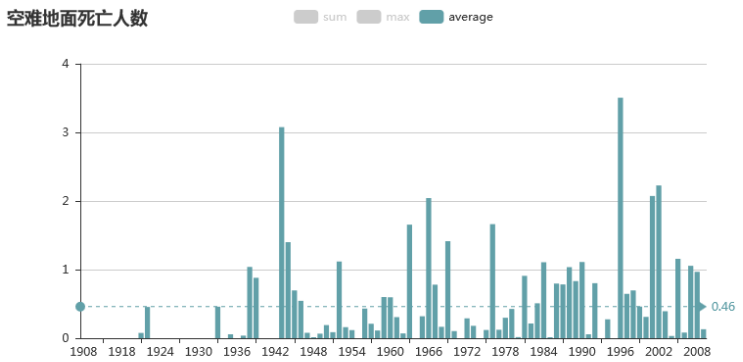
1908-2009年地面空难伤亡总人数 ：8440
911事件：恐怖袭击，地面死亡2750人
1996 Air Africa crash：飞机起飞速度不够，冲进了一个商场，地面死亡225人

Ground(去除911事件)





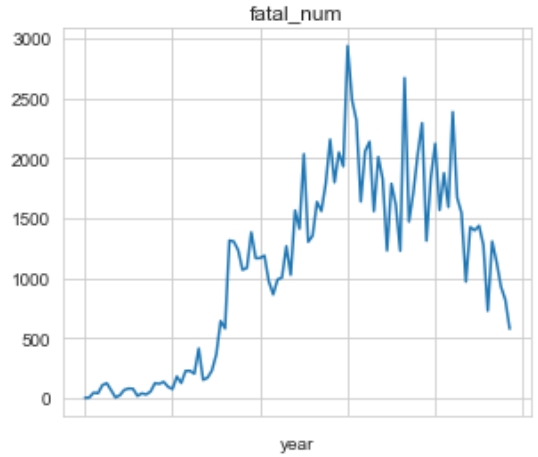


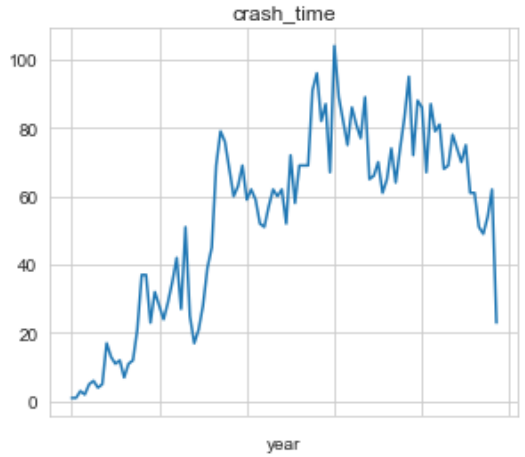


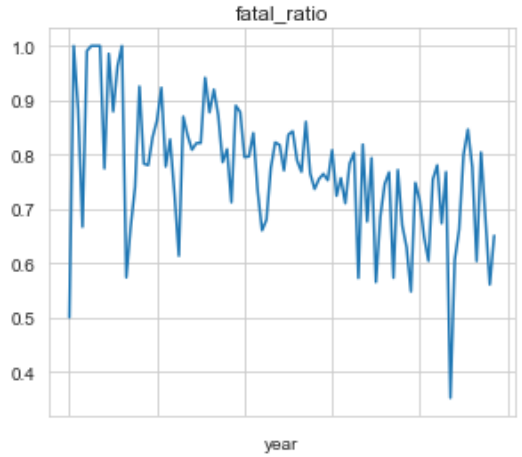
从上到下，分别描述了：每年地面伤亡总人数、每年地面伤亡最高人数和每年平均每次空难地面伤亡人数。

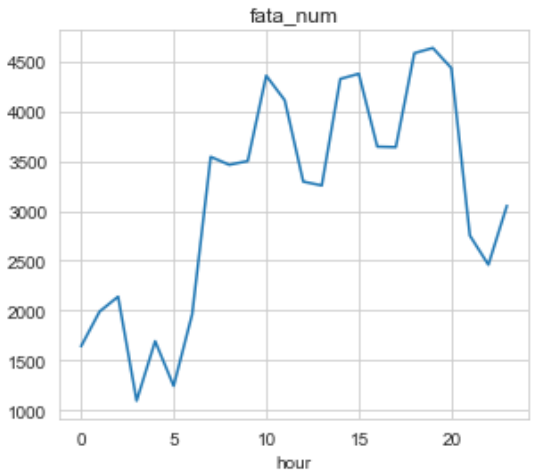
平均每次空难地面伤亡人数为0.46人。

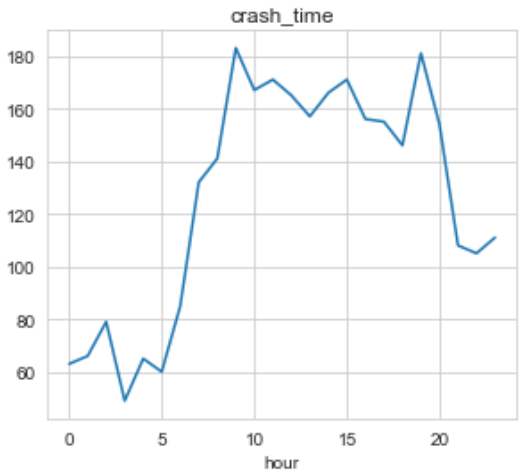
Fatalites

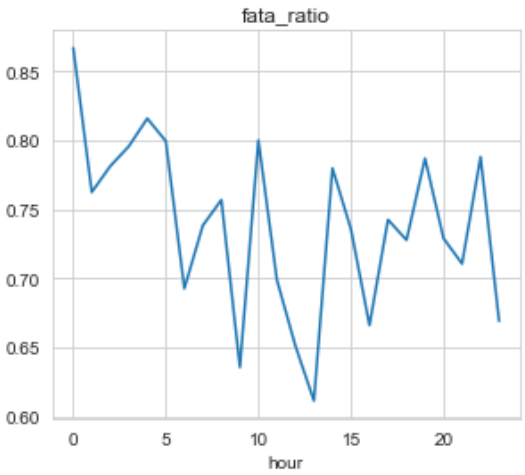










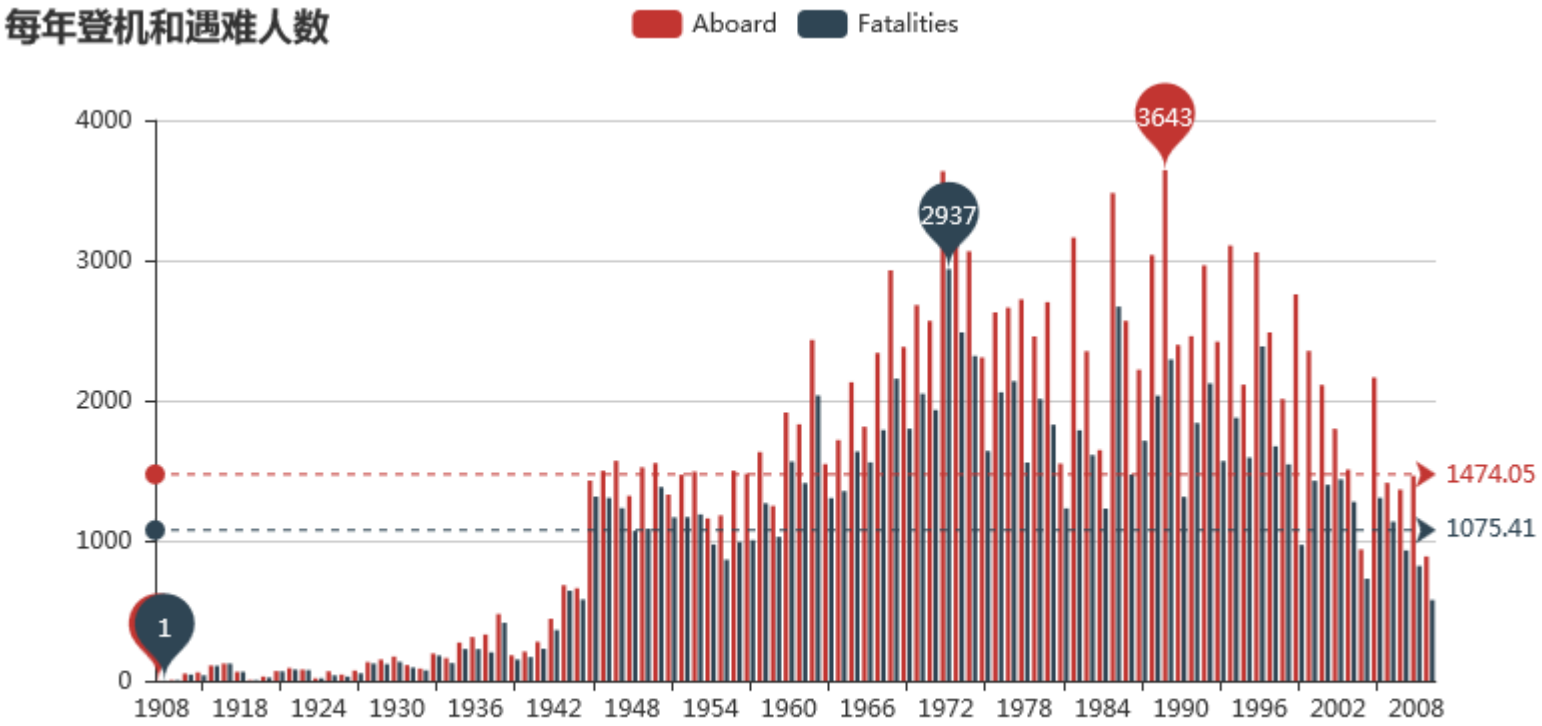


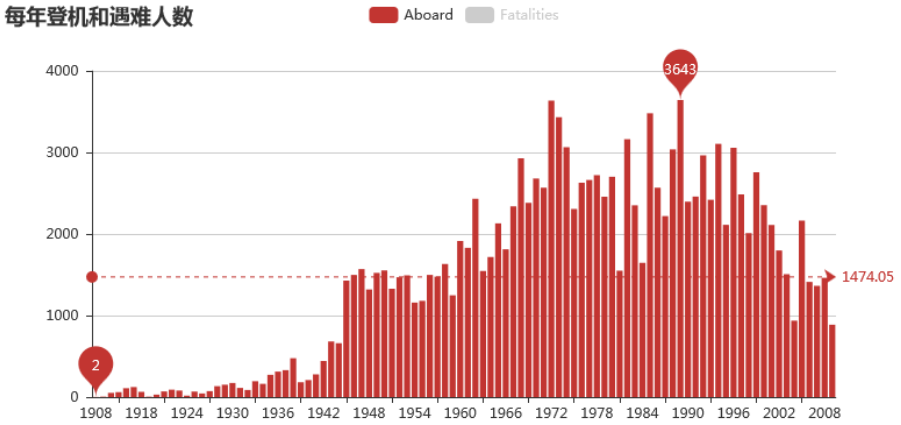
每年伤亡总人数、空难次数和伤亡率情况。

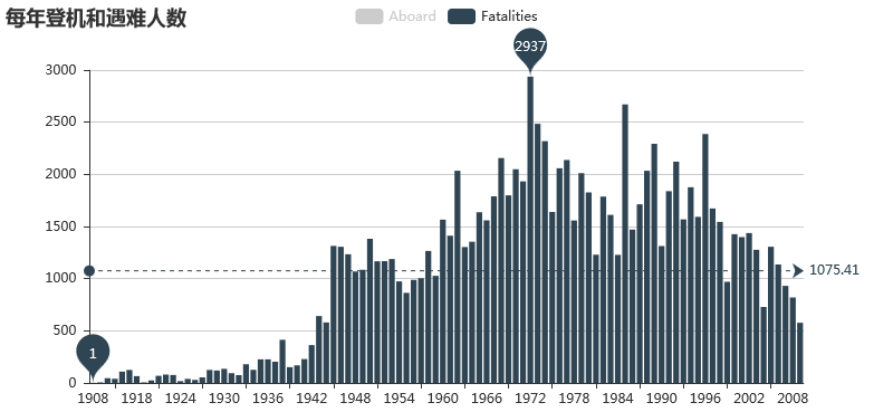
每小时伤亡总人数、空难次数和伤亡率情况。

借鉴博文：https://www.jianshu.com/p/4e9ad6aba1c8

每年登机和遇难人数





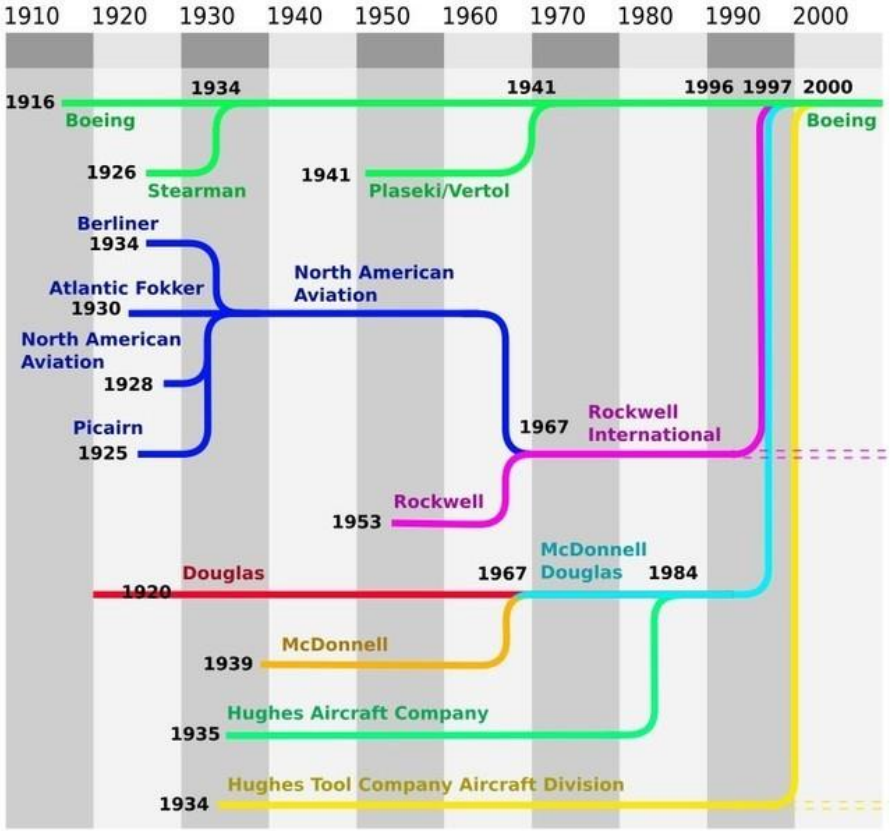


平均每年登机1474人，平均每年空难死亡1075人，发生空难乘客不易生还。

机型

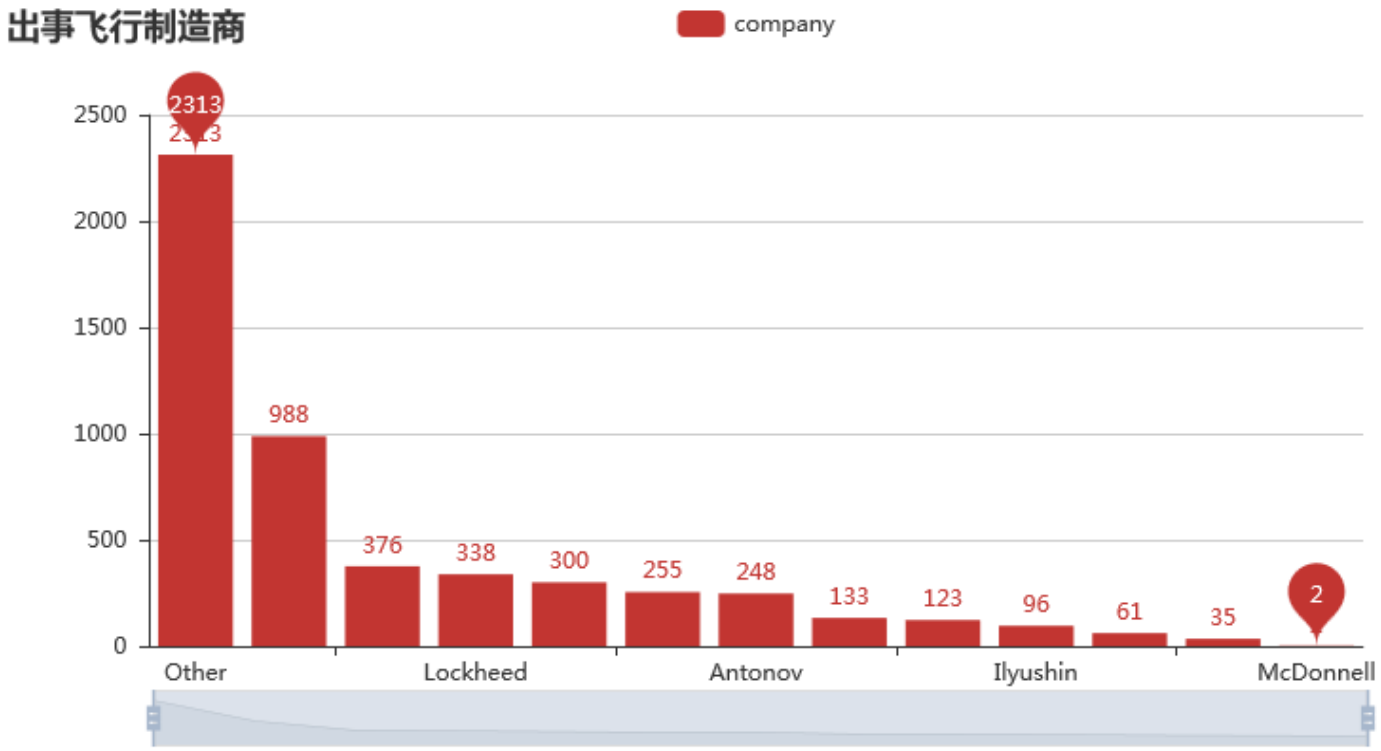
飞机制造厂商选取标准：1) 有名，百度百科可以查到；2) 出现次数较多，大于30次。

0	Other	其他	2313
1	Douglas	道格拉斯-美国	988
2	Boeing	波音-美国	376
3	Lockheed	洛克希德-美国	338
4	Cessna	塞斯纳-苏联	300
5	de Havilland	德.哈维尔-英国	255
6	Antonov	安东诺夫-法国	248
7	Fokker	福克-荷兰	133
8	McDonnell Douglas	麦克唐纳.道格拉斯-美国	123
9	Ilyushin	伊尔-苏联	96
10	Embraer	巴西航空工业公司-巴西	61
11	Airbbus	空客-法国	35
12	McDonnell	麦克唐纳-美国	2

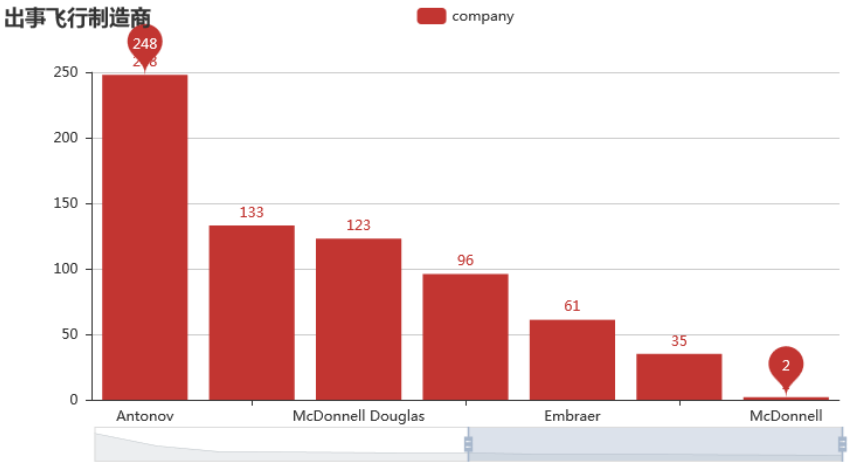


1916 波音成立；
1920 道格拉斯成立；
1939 麦克唐娜成立；
1967 麦克唐纳-道格拉斯 成立；
1997 M.D. 并入波音。

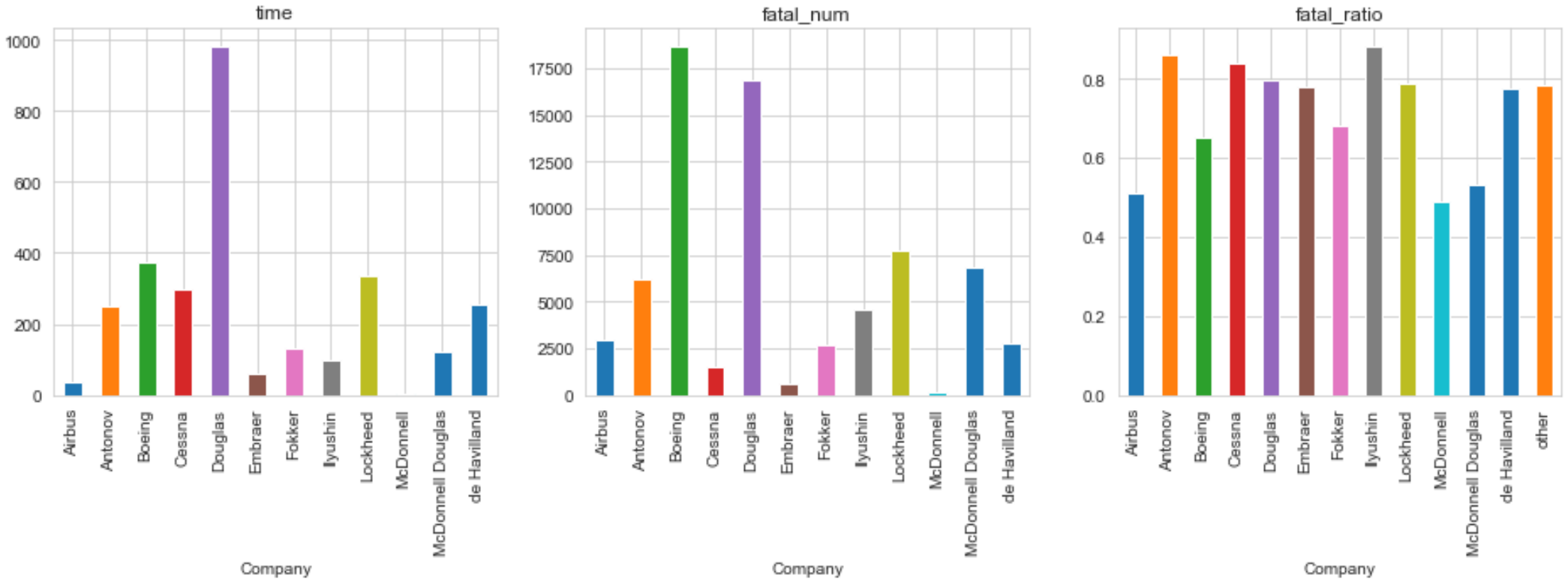
机型



0	Other	其他	2313
1	Douglas	道格拉斯-美国	988
2	Boeing	波音-美国	376
3	Lockheed	洛克希德-美国	338
4	Cessna	塞斯纳-苏联	300
5	de Havilland	德.哈维尔-英国	255
6	Antonov	安东诺夫-法国	248
7	Fokker	福克-荷兰	133
8	McDonnell Douglas	麦克唐纳.道格拉斯-美国	123
9	Ilyushin	伊尔-苏联	96
10	Embraer	巴西航空工业公司-巴西	61
11	Airbbus	空客-法国	35
12	McDonnell	麦克唐纳-美国	2



机型



各机型每年伤亡总人数、空难次数和伤亡率分布情况。

借鉴博文：https://www.jianshu.com/p/4e9ad6aba1c8

Summary

1 分句处理（前5句）

```
[['During a demonstration flight, a U.S. Army flyer flown by Orville Wright nose-dived into the ground from a height of approximately 75 fee
t, killing Lt. Thomas E. Selfridge who was a passenger.',
  'This was the first recorded airplane fatality in history.',
  'One of two propellers separated in flight, tearing loose the wires bracing the rudder and causing the loss of control of the aircraft.',
  'Orville Wright suffered broken ribs, pelvis and a leg.',
  'Selfridge suffered a crushed skull and died a short time later.'],
 ['First U.S. dirigible Akron exploded just offshore at an altitude of 1,000 ft. during a test flight.'],
 ['The first fatal airplane accident in Canada occurred when American barnstormer, John M. Bryant, California aviator was killed.']]
```

2 分词处理（前3个单词）

```
['During', 'a', 'demonstration']
```

3 小写处理（前3个单词）

```
['during', 'a', 'demonstration']
```

Summary

4 去除标点符号和停用词

```
demonstration/flight/u.s./army/flyer/flown/orville/wright/nose-dived/ground
```

5 词干化处理——动词去掉-ed, -ing等语态，名词去掉复数形式等

```
['demonstr', 'flight', 'u.s.', 'armi', 'flyer']
```

6 简单统计汇总

6.1 识别评论文本中常用固定词组搭配（上为词干化处理后，下为未进行词干化处理）

```
cargo plane; weather condit; short runway; attempt land; shortli take;
advers weather; burst flame; poor weather; plane crash; caught fire;
aircraft crash; loss control; midair collis; continu vfr; vfr flight;
emerg land; heavi rain; engin failur; land gear; final approach

cargo plane; attempting land; weather conditions; short runway;
shortly taking; adverse weather; poor weather; plane crashed; caught
fire; burst flames; emergency landing; loss control; aircraft crashed;
midair collision; landing gear; vfr flight; continued vfr; engine
failure; heavy rain; final approach
```

weather condit —— weather condition
attempt land —— attempting land
shortli take ——shortly taking
advers weather —— adverse weather

Summary

6.1 统计出现次数最多的前 20 个单词（左为词干化处理后，右为未进行词干化处理）

```
[('crash', 3458),
 ('aircraft', 2469),
 ('plane', 1913),
 ('land', 1373),
 ('pilot', 1348),
 ('engin', 1119),
 ('flight', 1063),
 ('approach', 1011),
 ('runway', 919),
 ('failur', 886),
 ('crew', 831),
 ('mountain', 775),
 ('attempt', 756),
 ("'s", 697),
 ('take', 647),
 ('control', 629),
 ('airport', 629),
 ('weather', 606),
 ('altitud', 602),
 ('condit', 597)]
```

```
[('crashed', 3214),
 ('aircraft', 2467),
 ('plane', 1863),
 ('pilot', 1198),
 ('flight', 1052),
 ('approach', 940),
 ('engine', 923),
 ('runway', 914),
 ('failure', 878),
 ('crew', 810),
 ('landing', 732),
 ("'s", 697),
 ('airport', 625),
 ('weather', 606),
 ('altitude', 598),
 ('mountain', 556),
 ('takeoff', 547),
 ('conditions', 540),
 ('taking', 534),
 ('land', 522)]
```

原有单词数：10193
词干化处理后单词数：8189

Summary 词云图


```
[('crashed', 3214),
 ('aircraft', 2467),
 ('plane', 1863),
 ('pilot', 1198),
 ('flight', 1052),
 ('approach', 940),
 ('engine', 923),
 ('runway', 914),
 ('failure', 878),
 ('crew', 810),
 ('landing', 732),
 ("'s", 697),
 ('airport', 625),
 ('weather', 606),
 ('altitude', 598),
 ('mountain', 556),
 ('takeoff', 547),
 ('conditions', 540),
 ('taking', 534),
 ('land', 522)]
```

动词
crashed、approach、landing、takeoff、taking、land
飞机
aircraft、plane、flight
飞机设备
engine
机场设备
runway、airport
人
pilot、crew
天气和其他
weather、condition、failure、altitude、mountain

遇难原因探究

天气

```
word_text_clear = Text(word_clear)
word_text_clear.similar('weather')
```

```
visibility icing judgement fog turbulence judgment ifr meteorological
foggy thunderstorm trees poor mountains instrument visiblity imc
whiteout flight ground aircraft
```

icing 雪, fog 雾, turbulence 海洋\天气等狂暴, thunderstorm 暴风雨, poor 恶劣的, whiteout 乳白天空（极地的一种大气光象）

地形

```
word_text_clear.similar('mountains')
```

```
sea burned mountain ocean ground trees field approach takeoff hill fog
mountainside jungle crashed taking water hillside approximately house
forest
```

fog, sea, ocean, ground, trees, hill, mountainside, hillside, approximately, house

失败

```
word_text_clear.similar('fail')
```

```
engine resulted height separated control immediately take section
caused collapse failed engines takeoff buckled dropped cargo well
rescued increased applied
```

engine 发动机, immediately 立即, collapse 撞击, takeoff 起飞

人员、沟通不顺、飞机故障等

词性标注

形容词个数： 15218
名词个数： 58581

出现次数较多的形容词

poor 357
due 311
short 302
low 288
high 228
engin 178
failur 168
right 166
tree 154
final 153
visual 141
crash 125
altitud 124
unknown 119
safe 119
mile 115
second 113
in-flight 112
undetermin 110
minimum 105

出现次数较多的名词

crash 3299
aircraft 2425
plane 1879
pilot 1325
land 1287
flight 1063
approach 981
engin 820
mountain 751
attempt 695
runway 626
control 612
airport 599
condit 587
failur 577
crew 501
ground 482
fire 447
caus 434
rout 426

二、UCL二分类数据集——credit6000

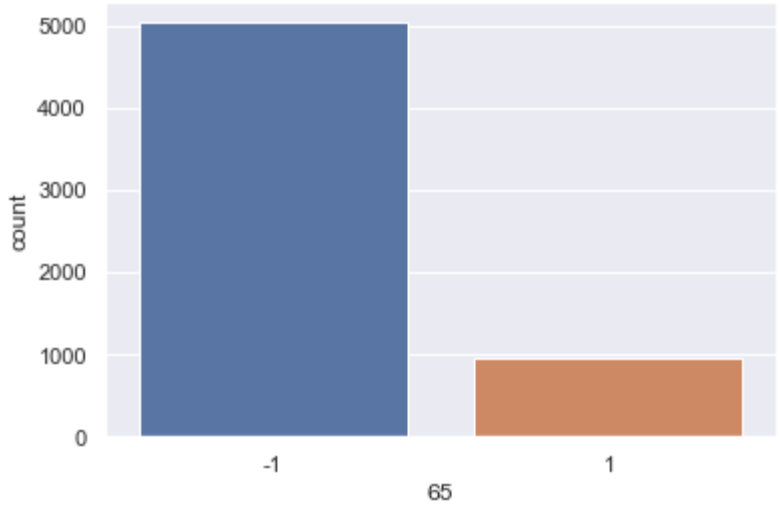
credit.head( ) credit.info( ) credit.describe( ) 函数进行数据初探

```
credit.head()
```

	0	1	2	3	4	5	6	7	8	9	...	56	57	58	59	60	61	62	63	64	65
0	8.4386	0.70572	130.24	1.3021	4.6052	4.7593	4.6052	0.0000	4.6052	0.0000	...	4.3531	1.0346	3.4012	0.0	0	0	2.0898	6.2893	1.0000	1
1	8.4251	0.00000	31.16	1.3596	4.6052	4.5884	4.6052	0.0000	5.2983	0.0000	...	2.9957	0.5000	4.6052	0.0	0	0	2.1450	6.7921	1.0000	1
2	7.6202	1.31520	0.00	2.0334	4.0372	3.7534	4.4427	0.0000	0.0000	0.0000	...	2.9957	0.5000	4.4998	0.0	0	0	2.2959	6.3045	1.0000	1
3	8.0736	2.51090	0.00	2.1634	4.5072	4.5200	4.2047	0.0000	4.2195	0.0000	...	8.6981	4.1945	0.0000	0.0	0	0	2.1638	5.8942	7.7401	1
4	6.7745	3.71360	0.00	2.3858	3.0445	3.1355	0.0000	2.9443	0.0000	2.9444	...	8.0146	4.3653	0.0000	0.0	2	0	2.3026	4.1733	5.4994	1

```
credit.describe()
```

	0	1	2	3	4	5	6	7	8	9	...	56
count	6000.000000	6000.000000	6000.000000	6000.000000	6000.000000	6000.000000	6000.000000	6000.000000	6000.000000	6000.000000	...	6000.000000
mean	6.775281	1.945099	243.601963	3.087651	4.041039	4.181983	3.564094	0.127925	3.480633	0.137962	...	7.205984
std	1.554231	1.513040	450.860271	1.966704	1.967858	1.891913	2.282198	0.573166	2.294475	0.595292	...	1.961628
min	4.605200	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	2.995700
25%	4.994850	0.325777	0.000000	1.000000	3.588000	3.721700	0.000000	0.000000	0.000000	0.000000	...	6.214600
50%	7.094850	1.979350	18.768500	2.334800	4.480300	4.573450	4.317500	0.000000	4.275550	0.000000	...	7.899000
75%	8.127175	3.713600	276.572500	5.567800	5.216825	5.312600	5.106000	0.000000	5.062700	0.000000	...	8.770125
max	9.624500	3.713600	2000.000000	5.567800	6.907800	6.907800	6.907800	2.995600	6.907800	2.999700	...	10.596000

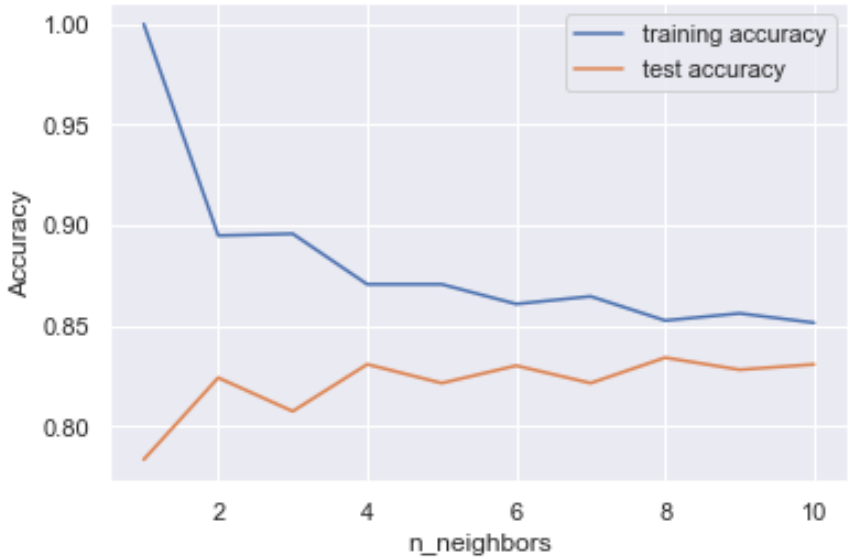


取值	数目	比例
-1	5040	84%
1	960	16%

LR

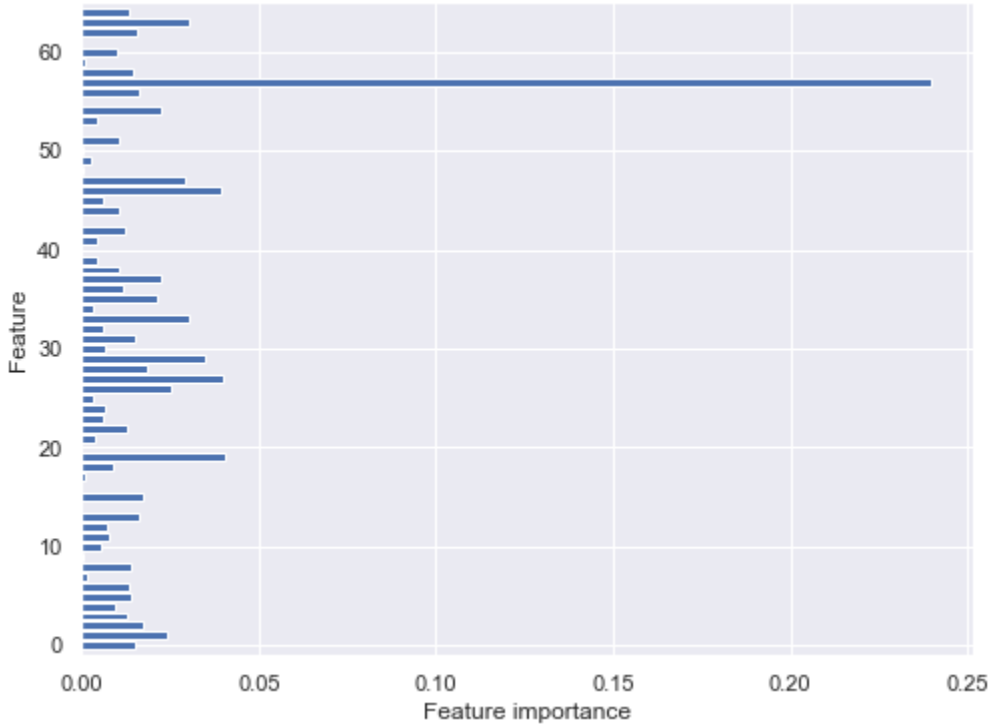
```
# LR Algorithm
# LogisticRegression和LogisticRegressionCV的主要区别是LogisticRegressionCV使用了交叉验证来选择正则化系数C。
# 而LogisticRegression需要自己每次指定一个正则化系数，且其默认带了正则化项。
# L1: solver='liblinear' 梯度下降，小数据集
# L2: solver='liblinear','sag','newton-cg','lbfgs' 前两个梯度下降，后两个牛顿法，sag适用于超大数据集
# multi_class='ovr' default，二元分类；multi_class='multinomial' 多元分类
# 样本是高度失衡的，class_weight='balanced',让类库自动提高非法用户样本的权重
# C为正则化系数λ的倒数 default=1 （交叉验证就是 Cs）
```

KNN

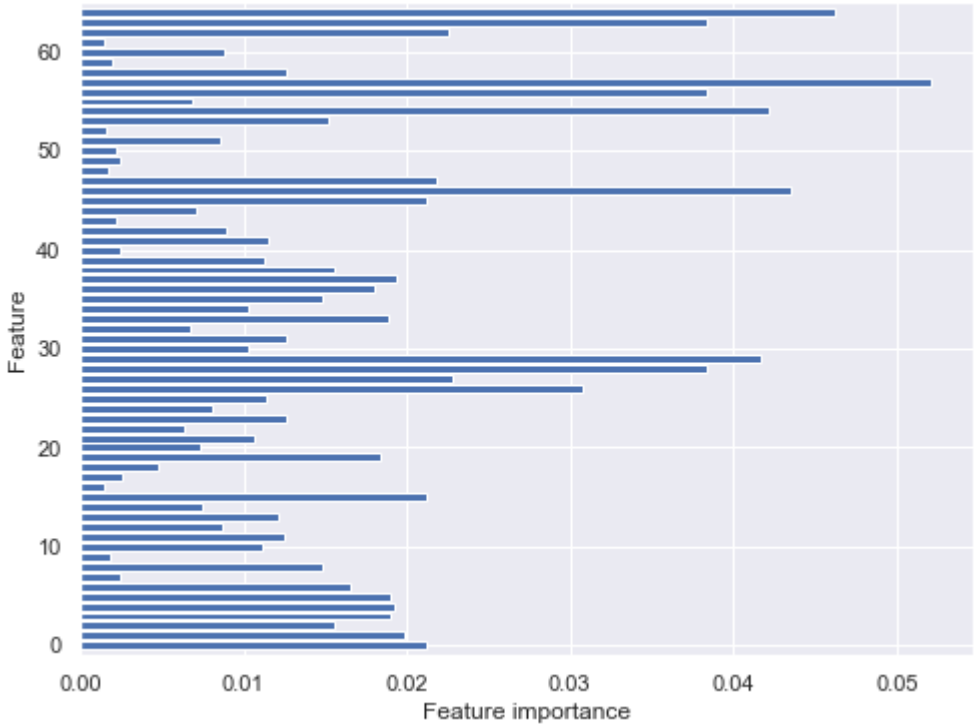


K = 9

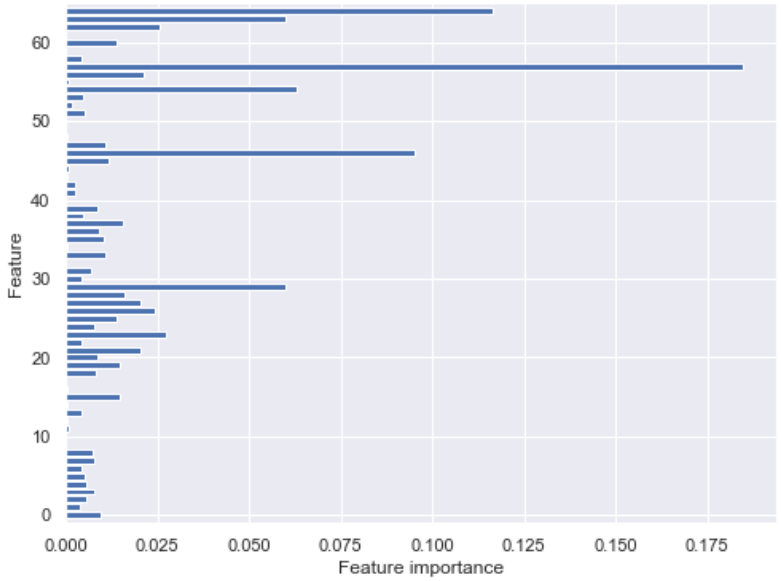
Decision Tree



RF



GBDT+GridSearchCV




```
GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=3,
              max_features=None, max_leaf_nodes=None,
              min_impurity_decrease=0.0, min_impurity_split=None,
              min_samples_leaf=1, min_samples_split=2,
              min_weight_fraction_leaf=0.0, n_estimators=100,
              n_iter_no_change=None, presort='auto', random_state=0,
              subsample=1.0, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
```

```
from sklearn.model_selection import GridSearchCV

param_test1 = {'n_estimators':range(10,200,5)}
gsearch1 = GridSearchCV(estimator = GradientBoostingClassifier(learning_rate=0.1,random_state=1),
                        param_grid = param_test1, scoring='accuracy',iid=False,cv=5)
gsearch1.fit(X_train, y_train)
gsearch1.best_params_,gsearch1.best_score_
# 最好的迭代次数为155步
```

```
({'n_estimators': 155}, 0.85)
```

```
from sklearn.model_selection import GridSearchCV

param_test2 = {'max_depth':range(3,10,1)}
gsearch2 = GridSearchCV(estimator = GradientBoostingClassifier(learning_rate=0.1,n_estimators=155,random_state=2),
                        param_grid = param_test2, scoring='accuracy',iid=False,cv=5)
gsearch2.fit(X_train, y_train)
gsearch2.best_params_,gsearch2.best_score_
# 最大树深为3
```

```
({'max_depth': 3}, 0.8495555555555556)
```

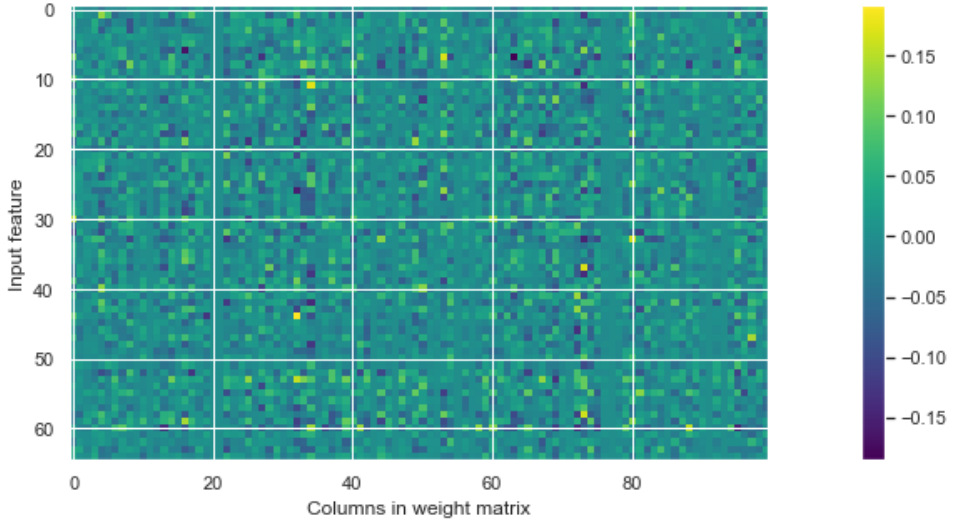
Neural Networks

首先数据变化均值为0，方差为1

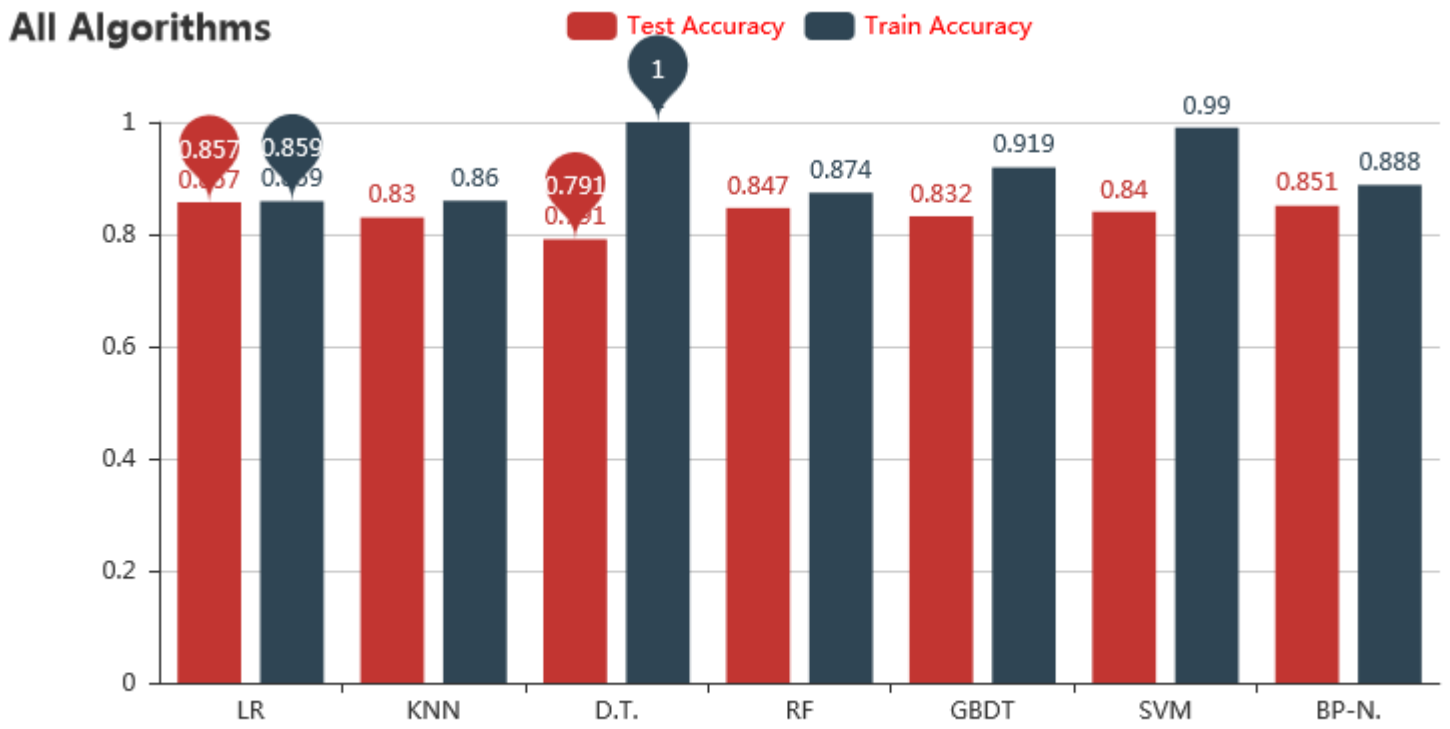
```
alpha = [i/10.0 for i in range(1,20)]
for k in range(0,19):
    mlp = MLPClassifier(max_iter=1000, alpha=alpha[k], random_state=0)
    mlp.fit(X_train_scaled, y_train)
    print("alpha: "+str(float(alpha[k])))
    print("Accuracy on training set: {:.3f}".format(mlp.score(X_train_scaled, y_train)))
    print("Accuracy on test set: {:.3f}".format(mlp.score(X_test_scaled, y_test)))
```

```
alpha: 0.8
Accuracy on training set: 0.887
Accuracy on test set: 0.853
alpha: 0.9
Accuracy on training set: 0.889
Accuracy on test set: 0.855
alpha: 1.0
Accuracy on training set: 0.876
Accuracy on test set: 0.847
```

调参：Alpha=0.9最优



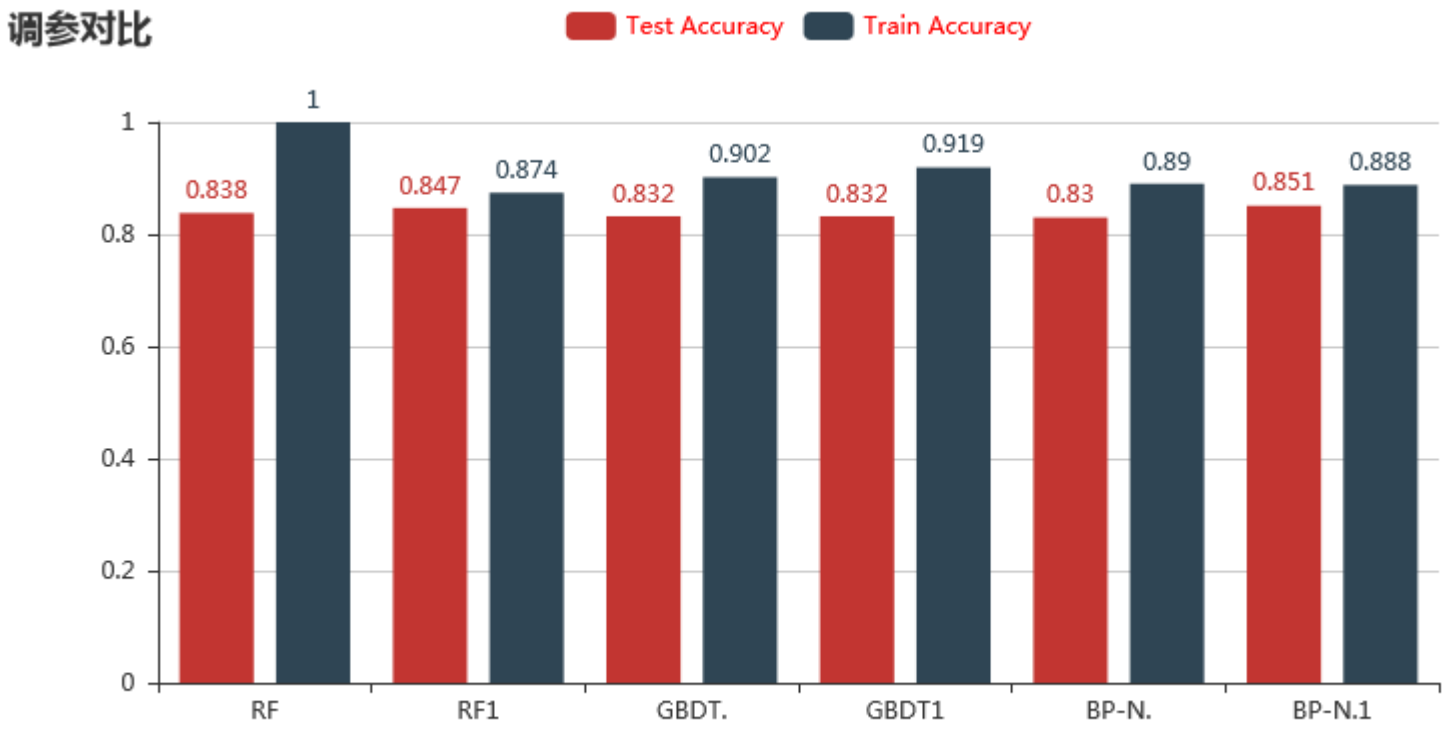
Summary



其中树决策树模型没有调参，过拟合，效果不好。

LR模型效果最好，准确率为0.857。

Summary——调参对比



其中RF过拟合了，增大树深max_depth和树的个数，准确率有所增加。

GBDT用GridSearchCV调参，最优树数目变为155，最大树深和默认值一致。

多层感知分类器，一般认为是线性模型的一种。用列表方式进行迭代调参，最优正则项系数Alpha=0.9。

三、Tweet数据集

描述性分析

总人数：6444人
特朗普：3218人支持
希拉里：3226人支持

评论总不同词数：16674
支持希拉里评论不同词数：10193
支持特普朗评论不同词数：9404

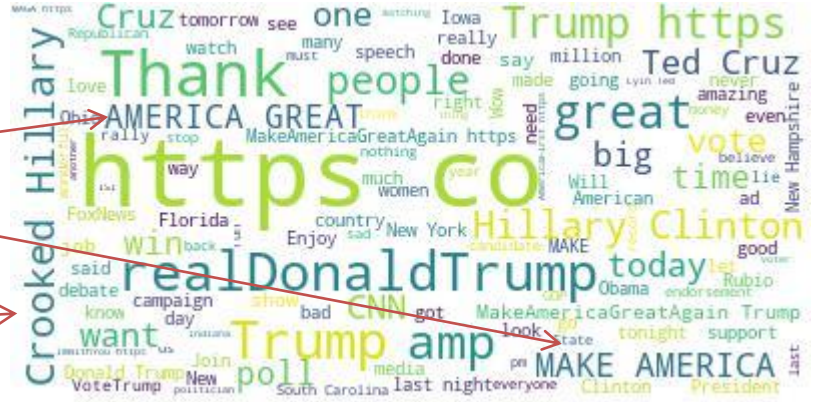
希拉里
克林顿

People 人民
America美国
Country国家

希拉里词云图

Make America Great
让美国再次复兴

Crooked Hillary
骗子希拉里



特朗普词云图

词袋模型

The question in this election: Who can put the plans into action that will make your life better? https://t.co/XreEY9OicG

共16674个词，去除标点

'The question in this election Who can put the plans into action that will make your life better https t co XreEY OicG'

建立词袋模型

[34, 1411, 8, 37, 231, 804, 32, 268, 4, 631, 228, 805, 19, 20, 70, 57, 252, 181, 3, 1, 2, 5755, 5756]

例如：标签 34 代表单词 The；
标签 1411 代表 question；
标签 4 代表 the 等。

词袋模型——Tensorflow

```
# 添加Embeding层
embed_size = 300
with graph.as_default():
    embedding = tf.Variable(tf.truncated_normal((n_words, embed_size), stddev=0.01))
    embed = tf.nn.embedding_lookup(embedding, inputs_)
```

词向量 Embedding

```
# 添加LSTM层
with graph.as_default():

    # 建立lstm层。这一层中，有 lstm_size 个 LSTM 单元
    lstm = tf.contrib.rnn.BasicLSTMCell(lstm_size)
     # 添加dropout
    drop = tf.contrib.rnn.DropoutWrapper(lstm, keep_prob)
    # 如果一层lsmt不够，多来几层
    cell = tf.contrib.rnn.MultiRNNCell([drop] * lstm_layers)
    # 对于每一条输入数据，都要有一个初始状态
    # 每次输入batch_size 个数据，因此有batch_size个初始状态
    initial_state = cell.zero_state(batch_size, tf.float32)
```

RNN
第一层： LSTM 256个序列
第二层： Dropout 舍弃概率0.8

词袋模型——Tensorflow

```
test_acc1 = []
with tf.Session(graph=graph) as sess:
    saver.restore(sess, tf.train.latest_checkpoint('checkpoints'))
    test_state = sess.run(cell.zero_state(batch_size, tf.float32))
    test_acc = float(math.pi/3.5)
    for ii, (x, y) in enumerate(get_batches(test_x, test_y, batch_size), 1):
        feed = {inputs_: x,
                labels_: y[:, None],
                keep_prob: 1,
                initial_state: test_state}
        batch_acc, test_state = sess.run([accuracy, final_state], feed_dict=feed)
        test_acc1.append(batch_acc)
    print("Test accuracy: {:.3f}".format(np.mean(test_acc)))
```

```
INFO:tensorflow:Restoring parameters from checkpoints\sentiment.ckpt
Test accuracy: 0.898
```

测试集准确率为0.898。

结果不是很理想，模型参数和函数还需调整。