1 빅데이터 응용 Final 프로젝트 과제 내용

빅데이터 응용 Final 프로젝트과제 2020-06-10 00:00:00 ~ 2020-06-26 23:59:00

- 다양한 데이터셋을 시각화하여 분석하는 프로젝트
- 평가방법 : 데이터의 시각화, 데이터의 분석 등
- 데이터셋을 하나 선정하여 창의적으로 여러방식으로 분석해보시기 바랍니다.
- Jupyter notebook file, readme file, pdf file, dataset file 네가지 압축하여 제출 (파일명: 학번_이름.zip)

Readme 파일: 데이터출처, 추가 패키지 설치 필요한 경우 설명 작성

pdf 파일: jupyter notebook 단순 변환

- 데이터셋 참조

https://brunch.co.kr/@data/10 (https://brunch.co.kr/@data/10)

https://brunch.co.kr/@jowlee/118 (https://brunch.co.kr/@jowlee/118)

- 프로젝트 예시 참조

Lecture 13,1410

Kaggle notebook (포함 온라인의 모든 자료들)

Contents of the Notebook:¶ Part1: Exploratory Data Analysis(EDA): 1)Analysis of the features.

2)Finding any relations or trends considering multiple features.

Part2: Feature Engineering and Data Cleaning: 1)Adding any few features.

2)Removing redundant features.

3)Converting features into suitable form for modeling.

Part3: Predictive Modeling 1)Running Basic Algorithms.

2)Cross Validation.

3)Ensembling.

4)Important Features Extraction.

2 문제 정의 및 데이터 수집

어떤 데이터를 골라야 할지 고민을 많이 했습니다. 타이타닉 데이터 처럼,

1. 데이터가 많아야 하고,
2. 이거 해볼만하겠는데? 라는 기준으로

https://brunch.co.kr/@jowlee/118 (https://brunch.co.kr/@jowlee/118)

위 에서 제공하는 다양한 데이터를 조회하였습니다.

그중에 제가 마음에 든것은 바로

도로교통공단에서 제공하는 교통사고 분석 시스템이였습니다. http://taas.koroad.or.kr/ (http://taas.koroad.or.kr/)

위에서 저는 http://taas.koroad.or.kr/api/selectDeathDataSet.do (http://taas.koroad.or.kr/api/selectDeathDataSet.do)

사고별 사망자 정보 csv 파일을 얻을 수 있었습니다.

In [92]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

executed in 7ms, finished 15:31:32 2020-06-26

In [93]:

```
import seaborn as sns
plt.style.use('fivethirtyeight')
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

executed in 10ms, finished 15:31:32 2020-06-26

In [94]:

```
import matplotlib
import matplotlib.font_manager as fm
fm.get_fontconfig_fonts()

#https://m.blog.naver.com/PostView.nhn?blogId=wideeyed&logNo=221225208497&proxyReferer=https:
# 위를 참고하여 한글 깨짐 해결.

font_location = './NanumFontSetup_TTF_SQUARE/NanumSquareB.ttf' # For Windows
font_name = fm.FontProperties(fname=font_location).get_name()
matplotlib.rc('font', family=font_name)
```

executed in 10ms, finished 15:31:32 2020-06-26

In [95]:

```
#data=pd.read_csv('12_19_death.csv',encoding = 'utf-8')
# 위 코드는 오류가 났다. 오류 메세지는
#'utf-8' codec can't decode byte 0xb9 in position 0: invalid start byte
# 위와 같았고
# 자세히 알아보니, file의 encoding 방식이 다른 것.
# 그래서 저는 excel을 이용해  12_19_death.csv를 csv-utf-8로 저장함으로써 해결하였습니다.

data=pd.read_csv('12_19_death_utf8.csv',encoding = 'utf-8')
```

executed in 597ms, finished 15:31:33 2020-06-26

In [96]:

```
#importing_test=pd.read_csv('C:/Users/like_/NaverCloud/2020년/학교 수업/빅데이터 응용/SourceC
```

executed in 6ms, finished 15:31:33 2020-06-26

3 Exploratory Data Analysis(EDA):

3.1 데이터 전처리

In [97]:

```
1 data.head()
```

executed in 71ms, finished 15:31:33 2020-06-26

Out[97]:

	발생년	발생년월일시	발생분	주야	요일	사망자수	사상자수	중상자수	경상자수	부상신고자수	...	당사자종별_1당_대분류	당사자종별_1당	당사자종별_2당_대분류	당사자종별_2당	발생위치X_UTMK	발생위치Y_UTMK	경도	
0	2012	2012010101	50	야간	일	1	1	0	0	0	...	승용차	중형	보행자	보행자	949860	949860	126.931891	37.(

3.1.1 몇건의 데이터, 그리고 column의 개수는

In [98]:

```
1 data.shape
```

executed in 11ms, finished 15:31:33 2020-06-26

Out[98]:

(34145, 29)

3만 4천개의 데이터, 그리고 column의 개수는 29개 였습니다.

3.1.2 column은 무엇이 있나

In [99]:

```
1 data.columns
```

executed in 11ms, finished 15:31:33 2020-06-26

Out[99]:

```
Index(['발생년', '발생년월일시', '발생분', '주야', '요일', '사망자수', '사상자수',
'중상자수', '경상자수',
       '부상신고자수', '발생지시도', '발생지시군구', '사고유형_대분류', '사고유형_중
분류', '사고유형', '법규위반_대분류',
       '법규위반', '도로형태_대분류', '도로형태', '당사자종별_1당_대분류', '당사자종
별_1당', '당사자종별_2당_대분류',
       '당사자종별_2당', '발생위치X_UTMK', '발생위치Y_UTMK', '경도', '위도', 'Unname
d: 27',
       'Unnamed: 28'],
      dtype='object')
```

3.1.3 의미 없는 컬럼의 삭제

발생위치X_UTMK 발생위치Y_UTMK 경도 위도 데이터는 알아봤자 잘 활용을 못할 거 같아서 지우겠습니다.

또한 Unnamed: 27, Unnamed: 27 은 의미 없는 데이터가 들어 있어 삭제하겠습니다.

In [100]:

```
1 #data.drop(['발생위치X_UTMK', '발생위치Y_UTMK' , '경도' ,'위도'], axis =1)
2 data = data.drop(['발생위치X_UTMK', '발생위치Y_UTMK' , '경도' ,'위도','Unnamed: 27', 'Unnamed
```

executed in 29ms, finished 15:31:33 2020-06-26

In [101]:

```
1 data.head()
```

executed in 80ms, finished 15:31:33 2020-06-26

	발생년	발생년월일시	발생분	주야	요일	사망자수	사상자수	중상자수	경상자수	부상신고자수	...	차도통행중	차도통행중	운전자법규위반	안전무시	운전의불이행	단일로	기타단일로	승용차	중형	보행자	보행자
0	2012	2012010101	50	야간	일	1	1	0	0	0	...											

3.2 결측치가 있나 확인

In [102]:

```
1 data.isnull().sum()
```

executed in 81ms, finished 15:31:33 2020-06-26

Out[102]:

```
발생년              0
발생년월일시           0
발생분              0
주야               0
요일               0
사망자수             0
사상자수             0
중상자수             0
경상자수             0
부상신고자수           0
발생지시도            0
발생지시군구           0
사고유형_대분류         0
사고유형_중분류         0
사고유형             0
법규위반_대분류         0
법규위반             0
```

다행스럽게 null 값이 하나도 없습니다.

저는 사고 유형을 주시했고, 어떤 값이 사고 유형에 영향을 끼쳤을 까 시각해 보고, 조사해보도록 하겠습니다.

3.3 사상자 수의 평균

In [103]:

```
1 data.describe()
```

executed in 142ms, finished 15:31:33 2020-06-26

Out[103]:

	발생년	발생년월일시	발생분	사망자수	사상자수	중상자수	34
count	34145.000000	3.414500e+04	34145.000000	34145.000000	34145.000000	34145.000000	
mean	2015.185152	2.015254e+09	27.219710	1.038922	1.608932	0.279865	
std	2.260179	2.260353e+06	17.396288	0.243548	2.042292	0.981340	
min	2012.000000	2.012010e+09	0.000000	1.000000	1.000000	0.000000	
25%	2013.000000	2.013092e+09	11.000000	1.000000	1.000000	0.000000	
50%	2015.000000	2.015080e+09	28.000000	1.000000	1.000000	0.000000	
75%	2017.000000	2.017082e+09	41.000000	1.000000	2.000000	0.000000	
max	2019.000000	2.019123e+09	59.000000	10.000000	105.000000	54.000000	

사상자 수는 평균 2명이였습니다. 그리고 최소 1명, 그리고 최대 105명이였음을 알수 있습니다.

In [104]:

```
1 data['사상자수'].value_counts()
```

executed in 20ms, finished 15:31:33 2020-06-26

Out[104]:

```
1     25284
2      4772
3      1854
4       926
5       495
6       247
7       151
8        98
9        68
11       38
10       38
12       31
13       17
15       14
14       14
18       13
19       10
```

In [105]:

```
1 data[data['사상자수']>=3].shape
```

executed in 24ms, finished 15:31:33 2020-06-26

Out[105]:

```
(4089, 23)
```

사상자 수 3명 이상은 너무 적습니다. 그래서 새로운 분류를 밑에서 만들어 보겠습니다.

3.3.1 사상자수로 부터 새로운 컬럼 사상자 분류를 만들고, 3명이상의 사상자는 3분류로 만들기

In [106]:

```
1 data['사상자분류']=data['사상자수']
2 data.loc[data['사상자분류']>=3, '사상자분류']=3
```

executed in 36ms, finished 15:31:33 2020-06-26

In [107]:

```
1 data.head()
```

executed in 85ms, finished 15:31:33 2020-06-26

Out[107]:

	발생년	발생년월일시	발생분	주야	요일	사망자수	사상자수	중상자수	경상자수	부상신고자수	...	사고유형	법규위반_대분류	법규위반	도로형태_대분류	도로형태	당사자종별_1당_대분류	당사자종별_1당	당사자종별_2당_대분류	당사자종별_2당
0	2012	2012010101	50	야간	일	1	1	0	0	0	...	차도통행중	운전자법규위반	안전운전불이행	단일로	기타단일로	승용차	중형	보행자	보행자
1	2012	2012010101	5	야간	일	1	6	5	0	0	...	정면충돌	운전자법규위반	중앙선침범	단일로	기타단일로	승용차	중형	승용차	소형
2	2012	2012010108	50	주간	일	1	1	0	0	0	...	공작물충돌	운전자법규위반	안전운전불이행	단일로	기타단일로	승용차	소형	없음	없음
3	2012	2012010110	25	주간	일	2	2	0	0	0	...	측면충돌	운전자법규위반	제한속도위반(20KM초과시)	교차로	교차로내	승합차	대형	승용차	소형
4	2012	2012010103	30	야간	일	1	1	0	0	0	...	도로이탈추락	운전자법규위반	안전운전불이행	단일로	기타단일로	승용차	중형	없음	없음

5 rows × 24 columns

3.4 사고 유형 대분류의 분포

In [108]:

```
1 data['사고유형_대분류'].value_counts()
```

executed in 26ms, finished 15:31:33 2020-06-26

Out[108]:

```
차대차        13484
차대사람       13367
차량단독        7285
철길건널목          9
Name: 사고유형_대분류, dtype: int64
```

In [109]:

```
1 data.사고유형_대분류.unique()
```

executed in 17ms, finished 15:31:34 2020-06-26

Out[109]:

```
array(['차대사람', '차대차', '차량단독', '철길건널목'], dtype=object)
```

차대사람, 차대차, 차량단독, 철길건널목 총 4개의 데이터가 있었습니다. 비율은 어떠할까요?

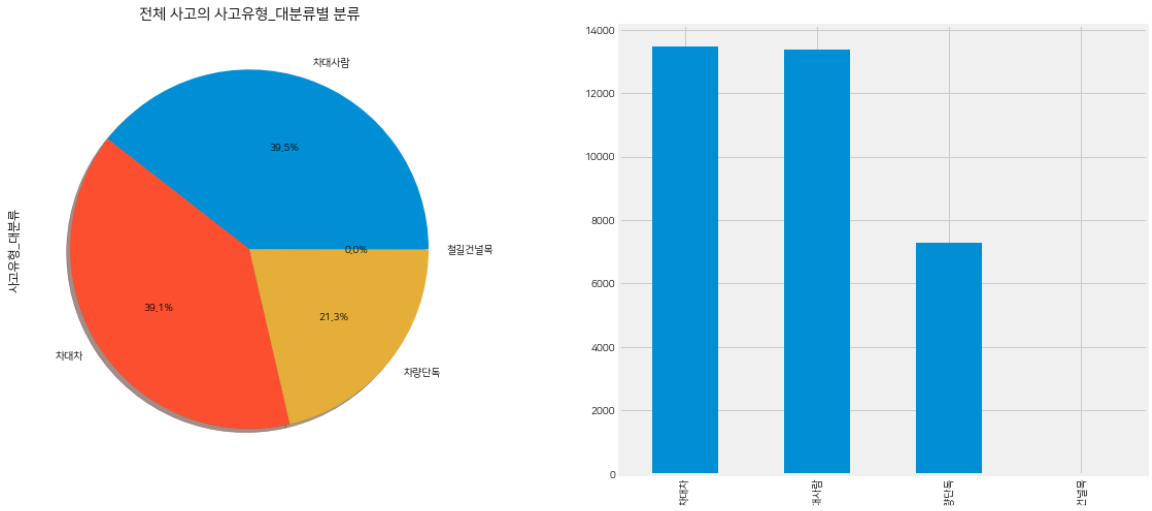
In [110]:

```
f,ax=plt.subplots(1,2,figsize=(18,8))
labels = list(data.사고유형_대분류.unique())
#data['사고유형_대분류'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0],s
data['사고유형_대분류'].value_counts().plot.pie(autopct='%1.1f%%', labels=labels,ax=ax[0],sha

#data['사고유형_대분류'].value_counts().plot
data['사고유형_대분류'].value_counts().plot.bar(ax=ax[1])
ax[0].set_title('전체 사고의 사고유형_대분류별 분류')

```

executed in 982ms, finished 15:31:34 2020-06-26

Out[110]:

Text(0.5, 1.0, '전체 사고의 사고유형_대분류별 분류')



3.4.1 너무 적은 데이터 철길건널목 삭제

In [111]:

```
idx_numbers = data[data['사고유형_대분류'] == '철길건널목'].index

idx_numbers
```

executed in 24ms, finished 15:31:35 2020-06-26

Out[111]:

```
Int64Index([385, 3402, 6228, 8850, 18705, 24726, 27680, 29864, 32554], dtype='int6
4')
```

In [112]:

```
data = data.drop(idx_numbers)
```

executed in 30ms, finished 15:31:35 2020-06-26

In [113]:

```
1 data['사고유형_대분류'].value_counts()
```

executed in 26ms, finished 15:31:35 2020-06-26

Out[113]:

```
차대차       13484
차대사람      13367
차량단독       7285
Name: 사고유형_대분류, dtype: int64
```

3.5 사상자 분류 에 따른 사고유형_대분류

In [114]:

```
1 data[data['사상자분류']==1].groupby(['사고유형_대분류'])['사고유형_대분류'].count()
```

executed in 49ms, finished 15:31:35 2020-06-26

Out[114]:

```
사고유형_대분류
차대사람      12341
차대차        7214
차량단독       5723
Name: 사고유형_대분류, dtype: int64
```

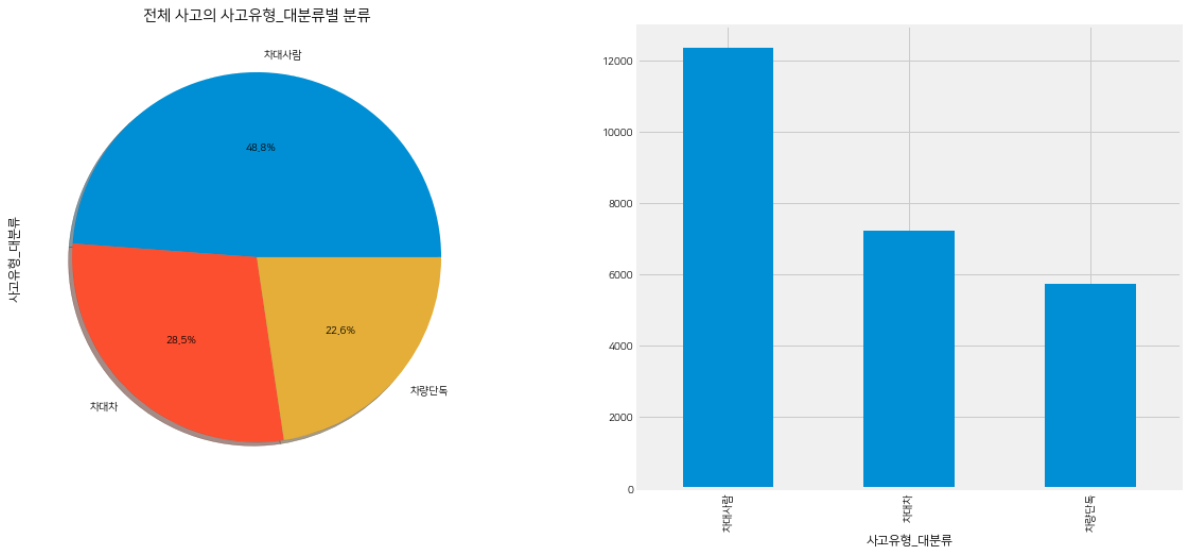
In [115]:

```
f,ax=plt.subplots(1,2,figsize=(18,8))
labels = list(data.사고유형_대분류.unique())
#data['사고유형_대분류'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0],s
data[data['사상자분류']==1].groupby(['사고유형_대분류'])['사고유형_대분류'].count().plot.pie(

#data['사고유형_대분류'].value_counts().plot
data[data['사상자분류']==1].groupby(['사고유형_대분류'])['사고유형_대분류'].count().plot.bar(
ax[0].set_title('전체 사고의 사고유형_대분류별 분류')

```

executed in 718ms, finished 15:31:35 2020-06-26

Out[115]:

Text(0.5, 1.0, '전체 사고의 사고유형_대분류별 분류')



In [116]:

```
data[data['사상자분류']==2].groupby(['사고유형_대분류'])['사고유형_대분류'].count()
```

executed in 24ms, finished 15:31:35 2020-06-26

Out[116]:

```
사고유형_대분류
차대사람     745
차대차     3063
차량단독     963
Name: 사고유형_대분류, dtype: int64
```

In [117]:

```
1 data[data['사상자분류']==3].groupby(['사고유형_대분류'])['사고유형_대분류'].count()
```

executed in 22ms, finished 15:31:35 2020-06-26

Out[117]:

```
사고유형_대분류
차대사람      281
차대차      3207
차량단독      599
Name: 사고유형_대분류, dtype: int64
```

차대사람분류가 사상자 1명인 케이스에서 제일 많았습니다. 그리고 사상자분류가 1,2인 케이스에서 차대차 분류가 가장 많았습니다.

3.6 사고유형 대분류 별 평균 사상자수

In [118]:

```
1 data[['사상자수','사고유형_대분류']].groupby(['사고유형_대분류']).mean()
```

executed in 33ms, finished 15:31:35 2020-06-26

Out[118]:

	사상자수
사고유형_대분류	
차대사람	1.117453
차대차	2.201053
차량단독	1.414825

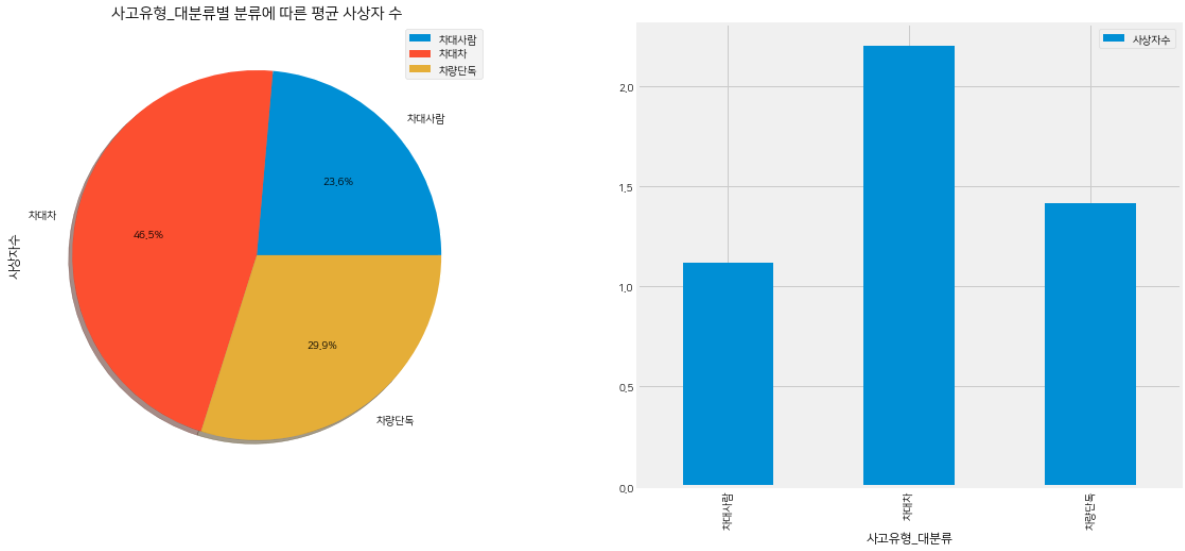
In [119]:

```
f,ax=plt.subplots(1,2,figsize=(18,8))
labels = list(data.사고유형_대분류.unique())
#data['사고유형_대분류'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0],s
data[['사상자수','사고유형_대분류']].groupby(['사고유형_대분류']).mean().plot.pie(autopct='%1

#data['사고유형_대분류'].value_counts().plot
data[['사상자수','사고유형_대분류']].groupby(['사고유형_대분류']).mean().plot.bar(ax=ax[1])
ax[0].set_title('사고유형_대분류별 분류에 따른 평균 사상자 수')

```

executed in 848ms, finished 15:31:36 2020-06-26

Out[119]:

Text(0.5, 1.0, '사고유형_대분류별 분류에 따른 평균 사상자 수')



아, 차대차가 사상자 평균 값이 높구나를 알수 있었습니다.

In [ ]:

```

```

executed in 39ms, finished 14:14:33 2020-06-26

3.7 주야에 따른 사상자 대분류

In [120]:

```
1 data.head()
```

executed in 88ms, finished 15:31:36 2020-06-26

	발생년	발생년월일시	발생분	주야	요일	사망자수	사상자수	중상자수	경상자수	부상신고자수	...	사고유형	법규위반_대분류	법규위반	도로형태_대분류	도로형태	당사자종별_1당_대분류	당사자종별_1당	당사자종별_2당_대분류	당사자종별_2당	사상자분류
0	2012	2012010101	50	야간	일	1	1	0	0	0	...	차도통행중	운전자법규위반	안전운전의무불이행	단일로	기타단일로	승용차	중형	보행자	보행자	1
1	2012	2012010101	5	야	일	1	6	5	0	0		정면	운전자법	중앙선	단일	기타단	승용	중	승용	소	3

In [121]:

```
1 data['주야분류']=data['주야']
```

executed in 12ms, finished 15:31:36 2020-06-26

In [122]:

```
1 data['주야분류'].value_counts()
```

executed in 42ms, finished 15:31:37 2020-06-26

Out[122]:

```
야간    17381
주간    16755
Name: 주야분류, dtype: int64
```

In [123]:

```
1 data.loc[data['주야분류']=='주간', '주야분류'] =0
```

executed in 43ms, finished 15:31:37 2020-06-26

In [124]:

```
1 data.loc[data['주야분류']=='야간', '주야분류'] =1
```

executed in 33ms, finished 15:31:37 2020-06-26

In [125]:

```
1 data.head()
```

executed in 98ms, finished 15:31:37 2020-06-26

Out[125]:

	발생년	발생년월일시	발생분	주야	요일	사망자수	사상자수	중상자수	경상자수	부상신고자수	...	법규위반_대분류	법규위반	도로형태_대분류	도로형태	당사자종별_1당_대분류	당사자종별_1당	당사자종별_2당_대분류	당사자종별_2당	사상자분류	주야분류
0	2012	2012010101	50	야간	일	1	1	0	0	0	...	운전자법규의	안전운전불이행	단일로	기타단일로	승용차	중형	보행자	보행자	1	1

In [126]:

```
1 data[data['주야분류']==0].groupby(['사고유형_대분류'])['사고유형_대분류'].count()
```

executed in 58ms, finished 15:31:37 2020-06-26

Out[126]:

```
사고유형_대분류
차대사람     5158
차대차      7818
차량단독     3779
Name: 사고유형_대분류, dtype: int64
```

주간에는 차대차 사고 비율이 많았습니다.

In [127]:

```
1 data[data['주야분류']==1].groupby(['사고유형_대분류'])['사고유형_대분류'].count()
```

executed in 57ms, finished 15:31:37 2020-06-26

Out[127]:

```
사고유형_대분류
차대사람     8209
차대차      5666
차량단독     3506
Name: 사고유형_대분류, dtype: int64
```

야간에는 차대사람 사고 비율이 높았습니다.

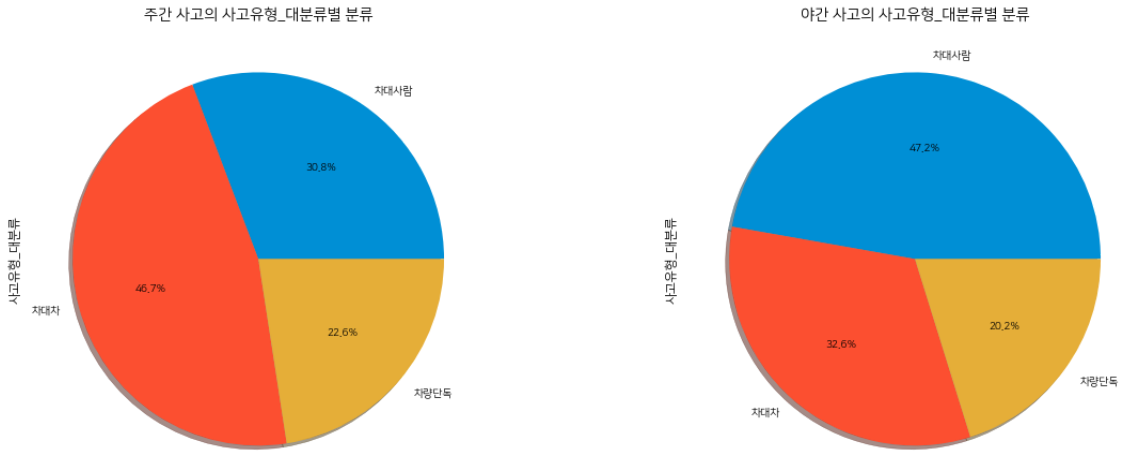
In [128]:

```
f,ax=plt.subplots(1,2,figsize=(18,8))
labels = list(data.사고유형_대분류.unique())
#data['사고유형_대분류'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0],s
data[data['주야분류']==0].groupby(['사고유형_대분류'])['사고유형_대분류'].count().plot.pie(au

#data['사고유형_대분류'].value_counts().plot
data[data['주야분류']==1].groupby(['사고유형_대분류'])['사고유형_대분류'].count().plot.pie(au
ax[0].set_title('주간 사고의 사고유형_대분류별 분류')
ax[1].set_title('야간 사고의 사고유형_대분류별 분류')

```

executed in 689ms, finished 15:31:38 2020-06-26

Out[128]:

Text(0.5, 1.0, '야간 사고의 사고유형_대분류별 분류')



주간에는 차대차, 야간에는 차대사람 사고의 비율이 제일 컸습니다.

3.8 요일에 따른 사상자 대분류 - 의미 없음.

In [129]:

```
data['요일'].value_counts()
```

executed in 23ms, finished 15:31:38 2020-06-26

Out[129]:

```
금    5121
월    5033
토    5027
화    4982
목    4878
수    4835
일    4260
Name: 요일, dtype: int64
```

In [130]:

```
1 oneTosevenList= list(range(7))
2 oneTosevenList
```

executed in 11ms, finished 15:31:38 2020-06-26

Out[130]:

```
[0, 1, 2, 3, 4, 5, 6]
```

In [131]:

```
1 data['요일'].replace(['일','월','화','수','목','금', '토'],oneTosevenList,inplace=True)
```

executed in 58ms, finished 15:31:38 2020-06-26

In [132]:

```
1 data.head()
```

executed in 65ms, finished 15:31:38 2020-06-26

Out[132]:

	발생년	발생년월일시	발생분	주야	요일	사망자수	사상자수	중상자수	경상자수	부상신고자수	...	법규위반_대분류	법규위반	도로형태_대분류	도로형태	당사자종별_1당_대분류	당사자종별_1당	당사자종별_2당_대분류	당사자종별_2당	사상자분류
0	2012	2012010101	50	야간	0	1	1	0	0	0	...	운전자법규위반	안전운전무이	단일로	기타단일로	승용차	중형	보행자	보행자	1
1	2012	2012010101	5	야간	0	1	6	5	0	0	...	운전자법규위반	중앙선침범	단일로	기타단일로	승용차	중형	승용차	소형	3
2	2012	2012010108	50	주간	0	1	1	0	0	0	...	운전자법규위반	안전운전무이	단일로	기타단일로	승용차	소형	없음	없음	1
3	2012	2012010110	25	주간	0	2	2	0	0	0	...	운전자법규위반	제한속도(20KM초과시)	교차로	교차로내	승합차	대형	승용차	소형	2
4	2012	2012010103	30	야간	0	1	1	0	0	0	...	운전자법규위반	안전운전무이	단일로	기타단일로	승용차	중형	없음	없음	1

5 rows × 25 columns

In [133]:

```
1 dateList = ['일','월','화','수','목','금', '토']
```

executed in 9ms, finished 15:31:38 2020-06-26

In [134]:

```
1 data[data['요일']==1]
```

executed in 154ms, finished 15:31:38 2020-06-26

Out[134]:

	발생년	발생년월일시	발생분	주야	요일	사망자수	사상자수	중상자수	경상자수	부상신고자수	...	법규위반_대분류	법규위반	도로형태_대분류	도로형태	당사자종별_1당_대분류	당사자종별_1당	당사자종별_2당_대분류	당사자종별_2당
6	2012	2012010210	0	주간	1	2	2	0	0	0	...	운전자법규위반	안전운전의무불이행	단일로	기타단일로	승용차	중형	없음	없음
13	2012	2012010218	20	야간	1	1	1	0	0	0	...	운전자법규위반	안전운전의무불이행	교차로	교차로부근	승합차	소형	보행자	보행자
14	2012	2012010205	30	야간	1	1	2	0	1	0	...	운전자법규위반	안전운전의무불이행	단일로	기타단일로	승용차	대형	승용차	대형
20	2012	2012010220	15	야간	1	1	1	0	0	0	...	운전자법규위반	안전운전의무불이행	단일로	횡단보도부근	승용차	소형	보행자	보행자

	발생년	발생년월일시	발생분	주야	요일	사망자수	사상자수	중상자수	경상자수	부상신고자수	...	법규위반_대분류	법규위반	도로형태_대분류	도로형태	당사자종별_1당_대분류	당사자종별_1당	당사자종별_2당_대분류	당사자종별_2당
32	2012	2012010213	35	주간	1	1	1	0	0	0	...	운전자법규위반	안전운전의무불이행	단일로	기타단일로	화물차	중형	없음	없음
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
34119	2019	2019123007	49	주간	1	1	2	0	1	0	...	운전자법규위반	중앙선침범	교차로	교차로내	승용차	승용차	승용차	승용차
34129	2019	2019112501	30	야간	1	1	1	0	0	0	...	운전자법규위반	안전운전의무불이행	단일로	교량위	승용차	승용차	없음	없음
34130	2019	2019120921	35	야간	1	1	4	0	0	3	...	운전자법규위반	안전운전의무불이행	교차로	교차로내	승용차	승용차	승용차	승용차

	발생년	발생년월일시	발생분	주야	요일	사망자수	사상자수	중상자수	경상자수	부상신고자수	...	법규위반_대분류	법규위반	도로형태_대분류	도로형태	당사자종별_1당_대분류	당사자종별_1당	당사자종별_2당_대분류	당사자종별_2당
34131	2019	2019121610	40	주간	1	1	1	0	0	0	...	운전자법규위반	신호또는지시에따를의무위반(정지선위반포함)	교차로	교차로내	이륜차	이륜차	화물차	화물차
34132	2019	2019122313	40	주간	1	1	1	0	0	0	...	운전자법규위반	안전운전의무불이행	기타	기타	승용차	승용차	보행자	보행자

5033 rows × 25 columns

In [135]:

```
for i in range(7):
    print(dateList[i], '요일의 사고유형 대분류')
    print(data[data['요일']==i].groupby(['사고유형_대분류'])['사고유형_대분류'].count())
    print('₩n---------₩n')
```

executed in 112ms, finished 15:31:38 2020-06-26

```
일 요일의 사고유형 대분류
사고유형_대분류
차대사람     1534
차대차      1529
차량단독     1197
Name: 사고유형_대분류, dtype: int64

---------

월 요일의 사고유형 대분류
사고유형_대분류
차대사람     1964
차대차      2035
차량단독     1034
Name: 사고유형_대분류, dtype: int64

---------

화 요일의 사고유형 대분류
```

대체적으로 요일에 따른 것은 의미가 없음을 알 수 있었습니다.

3.9 법규위반 대분류 에 따른 분류 - 의미 없음.

In [136]:

```
data.법규위반_대분류.unique()
```

executed in 19ms, finished 15:31:38 2020-06-26

Out[136]:

```
array(['운전자법규위반', '정비불량', '보행자과실'], dtype=object)
```

In [137]:

```
data['법규위반_대분류'].value_counts()
```

executed in 32ms, finished 15:31:38 2020-06-26

Out[137]:

```
운전자법규위반    34127
정비불량          8
보행자과실        1
Name: 법규위반_대분류, dtype: int64
```

정비 불량, 보행자 과실이 너무 적어 학습이 어려울 것으로 추측

3.10 지역에 따른 분류

In [138]:

```
1 data.발생지시도.unique()
```

executed in 18ms, finished 15:31:38 2020-06-26

Out[138]:

```
array(['서울', '전북', '충남', '경남', '경북', '전남', '충북', '부산', '경기', '인
천', '광주',
       '울산', '강원', '대구', '제주', '대전', '세종'], dtype=object)
```

In [139]:

```
1 areaList = list(data.발생지시도.unique())
2 areaList
```

executed in 20ms, finished 15:31:38 2020-06-26

Out[139]:

```
['서울',
 '전북',
 '충남',
 '경남',
 '경북',
 '전남',
 '충북',
 '부산',
 '경기',
 '인천',
 '광주',
 '울산',
 '강원',
 '대구',
 '제주',
 '대전',
 '세종']
```

In [140]:

```
for areaName in areaList:
    print(areaName, ' 지역의 사고유형 대분류')
    print(data[data['발생지시도']==areaName].groupby(['사고유형_대분류'])['사고유형_대분류'].
    print('₩n---------₩n')
```

executed in 359ms, finished 15:31:38 2020-06-26

```
서울  지역의 사고유형 대분류
사고유형_대분류
차대사람     1572
차대차       905
차량단독      294
Name: 사고유형_대분류, dtype: int64

---------

전북  지역의 사고유형 대분류
사고유형_대분류
차대사람      803
차대차      1027
차량단독      509
Name: 사고유형_대분류, dtype: int64

---------

충남  지역의 사고유형 대분류
```

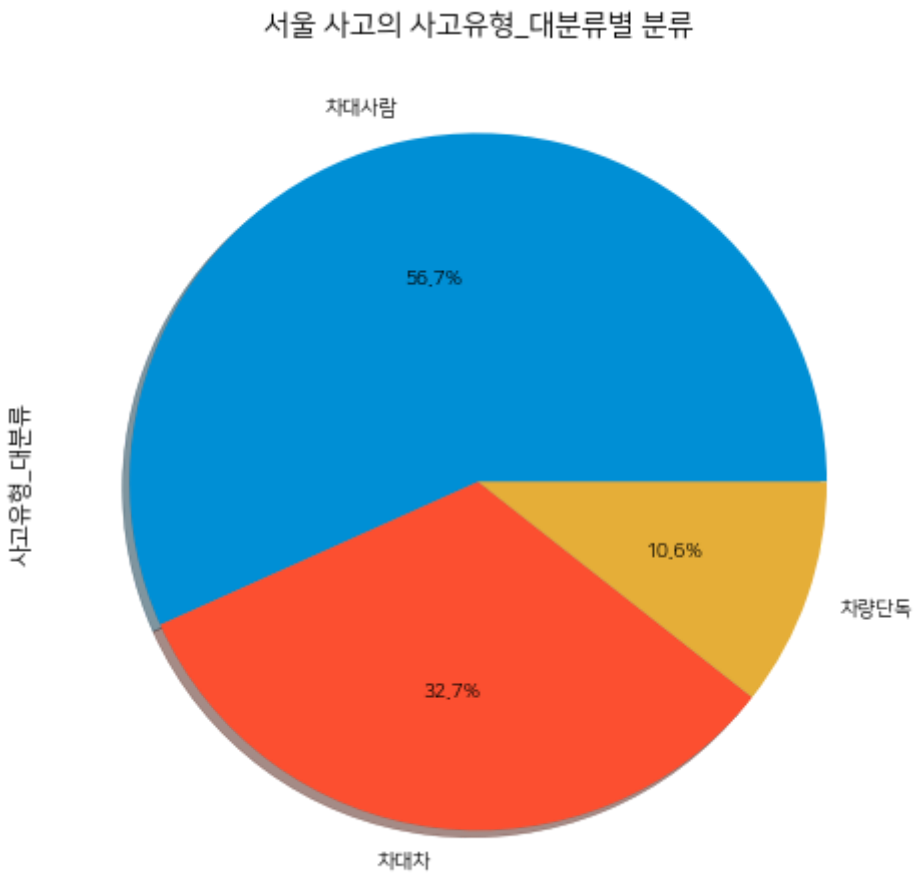
In [141]:

```
## 서울만 파이 플롯으로 그려보겠습니다!
f,ax=plt.subplots(1,1,figsize=(18,8))
labels = list(data.사고유형_대분류.unique())
data[data['발생지시도']=='서울'].groupby(['사고유형_대분류'])['사고유형_대분류'].count().plot
ax.set_title('서울 사고의 사고유형_대분류별 분류')

```

executed in 600ms, finished 15:31:39 2020-06-26

Out[141]:

Text(0.5, 1.0, '서울 사고의 사고유형_대분류별 분류')



차대 사람의 비율 이 높은 지역 : 서울, 광주, 제주

차대차 비율이 높은 지역 : 전북, 충남, 경북, 충북, 세종

비등비등한 지역 : 나머지.

로 알 수 있었습니다. 의미 있는 해석이 되었습니다..!

In [ ]:

```

```

3.11 도로형태_대분류에 따른 분류

In [142]:

```
1 data.도로형태_대분류.unique()
```

executed in 25ms, finished 15:31:39 2020-06-26

Out[142]:

```
array(['단일로', '교차로', '기타/불명', ' 횡단 후진등 금지위반(부당한 회전)', '어린
이보호위반', ' 최저속도',
       '보행자통행방해행위)', '유턴', '기타', '불명', '주차장'], dtype=object)
```

In [143]:

```
loadList = list(data.도로형태_대분류.unique())
loadList

for loadtype in loadList:
    print(loadtype, ' 형태의 도로의 사고유형 대분류')
    print(data[data['도로형태_대분류']==loadtype].groupby(['사고유형_대분류'])['사고유형_대분
    print('₩n---------₩n')
```

executed in 276ms, finished 15:31:39 2020-06-26

```
단일로  형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람    8670
차대차     7754
차량단독    5847
Name: 사고유형_대분류, dtype: int64

---------

교차로  형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람    4152
차대차     5257
차량단독    1095
Name: 사고유형_대분류, dtype: int64

---------

기타/불명  형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람    215
차대차     158
차량단독    185
Name: 사고유형_대분류, dtype: int64

---------

 횡단 후진등 금지위반(부당한 회전)  형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람     25
차대차     156
Name: 사고유형_대분류, dtype: int64

---------

어린이보호위반  형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람    36
차대차     34
차량단독    48
Name: 사고유형_대분류, dtype: int64

---------

 최저속도  형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람    13
차대차      3
차량단독     1
```

```
Name: 사고유형_대분류, dtype: int64

---------

보행자통행방해행위) 형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람    107
차대차       6
Name: 사고유형_대분류, dtype: int64

---------

유턴 형태의 도로의 사고유형 대분류
사고유형_대분류
차대차    9
Name: 사고유형_대분류, dtype: int64

---------

기타 형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람    136
차대차       99
차량단독    104
Name: 사고유형_대분류, dtype: int64

---------

불명 형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람    3
차대차      8
차량단독    1
Name: 사고유형_대분류, dtype: int64

---------

주차장 형태의 도로의 사고유형 대분류
사고유형_대분류
차대사람    10
차량단독     4
Name: 사고유형_대분류, dtype: int64

---------
```

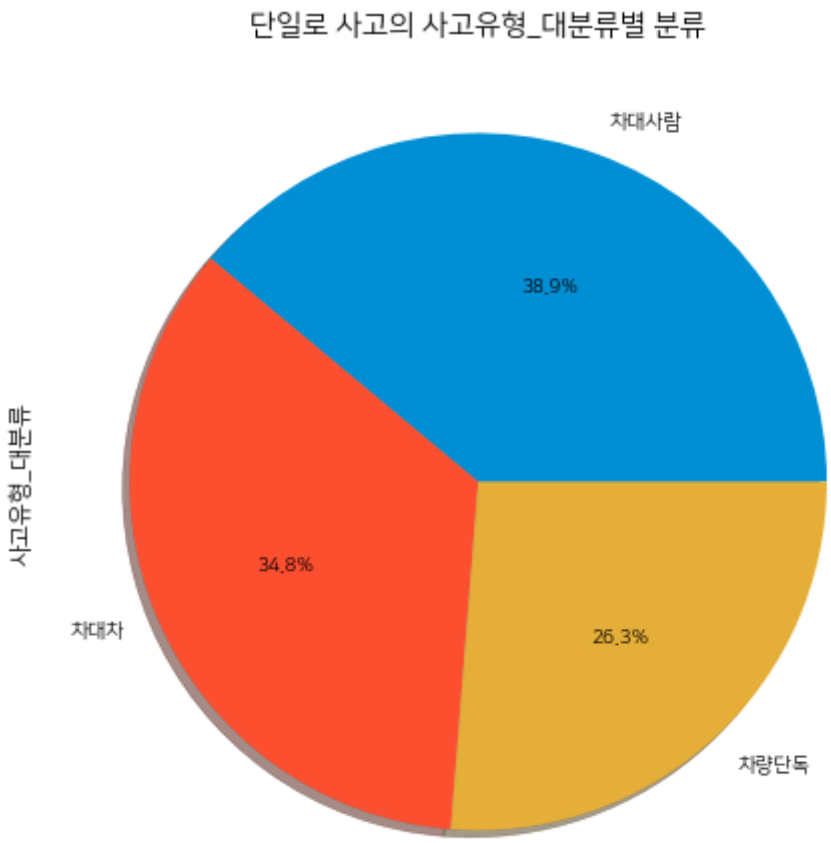
In [144]:

```
## 단일로만 파이 플롯으로 그려보겠습니다!
f,ax=plt.subplots(1,1,figsize=(18,8))
labels = list(data.사고유형_대분류.unique())
data[data['도로형태_대분류']=='단일로'].groupby(['사고유형_대분류'])['사고유형_대분류'].count
ax.set_title('단일로 사고의 사고유형_대분류별 분류')

```

executed in 591ms, finished 15:31:40 2020-06-26

Out[144]:

Text(0.5, 1.0, '단일로 사고의 사고유형_대분류별 분류')



결과를 보면, 차대사람 비율이 높은 경우 : 단일로, 기타/불명, 보행자 통행 방해 등등..

차대차 비율이 높은 경우 : 교차로, 등등..

차량 단독의 비율이 높은 경우 : 없음

으로 알 수 있었습니다.

3.12 당사자종별_1당_대분류 에 따른 분류

In [145]:

```
attackCarTypeList = list(data.당사자종별_1당_대분류.unique())
attackCarTypeList

for attackCarType in attackCarTypeList:
    print(attackCarType, ' 와 같은 가해자 차종에 따른 사고유형 대분류')
    print(data[data['당사자종별_1당_대분류']==attackCarType].groupby(['사고유형_대분류'])['사
    print('₩n---------₩n')
```

executed in 399ms, finished 15:31:40 2020-06-26

```
승용차  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    7911
차대차     5714
차량단독    2947
Name: 사고유형_대분류, dtype: int64

---------

승합차  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    1482
차대차      707
차량단독     207
Name: 사고유형_대분류, dtype: int64

---------

원동기장치자전거  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     67
차대차     712
차량단독    508
Name: 사고유형_대분류, dtype: int64

---------

이륜차  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     219
차대차     1605
차량단독    1352
Name: 사고유형_대분류, dtype: int64

---------

화물차  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    2960
차대차     3419
차량단독    1287
Name: 사고유형_대분류, dtype: int64

---------

농기계  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     19
차대차       77
```

```
차량단독     472
Name: 사고유형_대분류, dtype: int64

---------

특수차  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     109
차대차      178
차량단독      59
Name: 사고유형_대분류, dtype: int64

---------

자전거  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람      47
차대차      490
차량단독     189
Name: 사고유형_대분류, dtype: int64

---------

교차로부근  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     21
차대차      23
Name: 사고유형_대분류, dtype: int64

---------

건설기계  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     346
차대차      306
차량단독      98
Name: 사고유형_대분류, dtype: int64

---------

단일로  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     28
차대차      29
차량단독     35
Name: 사고유형_대분류, dtype: int64

---------

교차로내  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람      8
차대차      55
Name: 사고유형_대분류, dtype: int64

---------

기타단일로  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     94
차대차      59
```

```
Name: 사고유형_대분류, dtype: int64

---------

불명  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    24
차대차     27
차량단독     5
Name: 사고유형_대분류, dtype: int64

---------

횡단보도부근  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    2
차대차     3
Name: 사고유형_대분류, dtype: int64

---------

횡단보도상  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     3
차대차     17
Name: 사고유형_대분류, dtype: int64

---------

교차로  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     7
차대차      9
차량단독    11
Name: 사고유형_대분류, dtype: int64

---------

기타/불명  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    9
차대차     3
차량단독    1
Name: 사고유형_대분류, dtype: int64

---------

고가도로위  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    1
Name: 사고유형_대분류, dtype: int64

---------

교량위  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대차     1
차량단독    1
Name: 사고유형_대분류, dtype: int64

---------
```

```
사륜오토바이(ATV)  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     1
차대차     31
차량단독    98
Name: 사고유형_대분류, dtype: int64

---------

기타  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    5
차대차     7
차량단독    5
Name: 사고유형_대분류, dtype: int64

---------

개인형이동수단(PM)  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람     1
차대차      5
차량단독    10
Name: 사고유형_대분류, dtype: int64

---------

교차로횡단보도내  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    2
차대차     7
Name: 사고유형_대분류, dtype: int64

---------

지하차도(도로)내  와 같은 가해자 차종에 따른 사고유형 대분류
사고유형_대분류
차대사람    1
Name: 사고유형_대분류, dtype: int64

---------
```

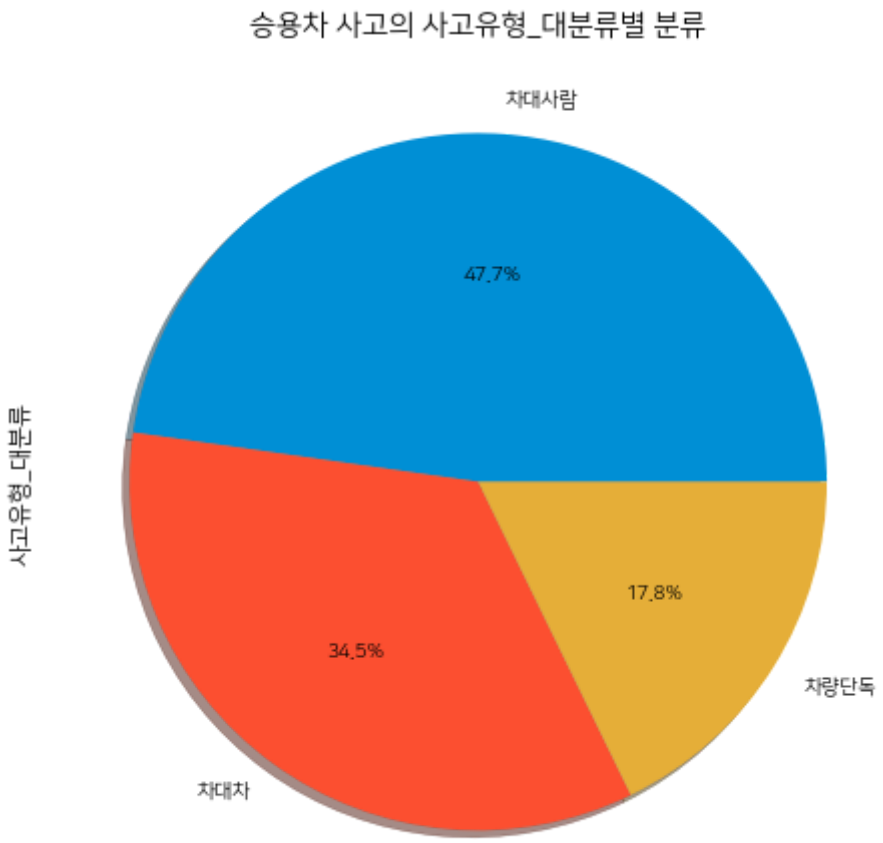
In [146]:

```
## 승용차 사고만 파이 플롯으로 그려보겠습니다!
f,ax=plt.subplots(1,1,figsize=(18,8))
labels = list(data.사고유형_대분류.unique())
data[data['당사자종별_1당_대분류']=='승용차'].groupby(['사고유형_대분류'])['사고유형_대분류']
ax.set_title('승용차 사고의 사고유형_대분류별 분류')

```

executed in 327ms, finished 15:31:41 2020-06-26

Out[146]:

Text(0.5, 1.0, '승용차 사고의 사고유형_대분류별 분류')



결과를 보면, 차대사람 비율이 높은 경우 : 승용차, 승합차,

차대차 비율이 높은 경우 : 원동기장치자전거, 화물차, 자전거,

차량 단독의 비율이 높은 경우 : 농기계

위와 같음을 알 수 있었습니다~!

4 Feature Engineering and Data Cleaning

4.1 필요없는 컬럼 제거하기

위 5가지 인풋과 아웃풋 1개를 제외한 나머지는 다 제거할 예정입니다.

In [147]:

```
1 data.columns
```

executed in 12ms, finished 15:31:41 2020-06-26

Out[147]:

```
Index(['발생년', '발생년월일시', '발생분', '주야', '요일', '사망자수', '사상자수',
'중상자수', '경상자수',
       '부상신고자수', '발생지시도', '발생지시군구', '사고유형_대분류', '사고유형_중
분류', '사고유형', '법규위반_대분류',
       '법규위반', '도로형태_대분류', '도로형태', '당사자종별_1당_대분류', '당사자종
별_1당', '당사자종별_2당_대분류',
       '당사자종별_2당', '사상자분류', '주야분류'],
      dtype='object')
```

In [148]:

```
1 column_set = set(data.columns)
```

executed in 7ms, finished 15:31:41 2020-06-26

In [149]:

```
1 wants_set = {'발생지시도', '사고유형_대분류','도로형태_대분류', '당사자종별_1당_대분류','사상
```

executed in 7ms, finished 15:31:41 2020-06-26

In [150]:

```
1 deleteIndexList = list(column_set.difference(wants_set))
```

executed in 9ms, finished 15:31:41 2020-06-26

In [151]:

```
1 deleteIndexList
```

executed in 11ms, finished 15:31:41 2020-06-26

Out[151]:

```
['사상자수',
 '법규위반',
 '사망자수',
 '경상자수',
 '사고유형_중분류',
 '주야',
 '중상자수',
 '요일',
 '발생년',
 '부상신고자수',
 '도로형태',
 '당사자종별_2당_대분류',
 '발생지시군구',
 '법규위반_대분류',
 '발생년월일시',
 '당사자종별_2당',
 '당사자종별_1당',
 '발생분',
 '사고유형']
```

In [152]:

```
1 data = data.drop(deleteIndexList, axis =1)
```

executed in 14ms, finished 15:31:41 2020-06-26

In [153]:

```
1 data
```

executed in 45ms, finished 15:31:41 2020-06-26

Out[153]:

	발생지시도	사고유형_대분류	도로형태_대분류	당사자종별_1당_대분류	사상자분류	주야분류
0	서울	차대사람	단일로	승용차	1	1
1	전북	차대차	단일로	승용차	3	1
2	충남	차량단독	단일로	승용차	1	0
3	경남	차대차	교차로	승합차	2	0
4	경북	차량단독	단일로	승용차	1	1
...	...	...	...	...	...	...
34140	경기	차대사람	보행자통행방해행위)	교차로내	3	1
34141	경기	차대차	단일로	승용차	1	1

4.2 명목형 값을 특정 숫자로 변환하기

4.2.1 발생지시도 값 먼저 변환해보기

In [154]:

```
#lookup_fruit_name = dict(zip(fruits.fruit_label.unique(), fruits.fruit_name.unique()))
lookup_Area_value = dict(zip(data.발생지시도.unique(), range(len(data.발생지시도.unique()))))
lookup_Area_value

```

executed in 25ms, finished 15:31:41 2020-06-26

Out[154]:

```
{'서울': 0,
 '전북': 1,
 '충남': 2,
 '경남': 3,
 '경북': 4,
 '전남': 5,
 '충북': 6,
 '부산': 7,
 '경기': 8,
 '인천': 9,
 '광주': 10,
 '울산': 11,
 '강원': 12,
 '대구': 13,
 '제주': 14,
 '대전': 15,
 '세종': 16}
```

In [155]:

```
lookup_Area_value.keys()
```

executed in 11ms, finished 15:31:41 2020-06-26

Out[155]:

```
dict_keys(['서울', '전북', '충남', '경남', '경북', '전남', '충북', '부산', '경기',
'인천', '광주', '울산', '강원', '대구', '제주', '대전', '세종'])
```

In [156]:

```
lookup_Area_value.values()
```

executed in 9ms, finished 15:31:41 2020-06-26

Out[156]:

```
dict_values([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16])
```

In [157]:

```
data['발생지시도'].replace(list(lookup_Area_value.keys()),list(lookup_Area_value.values()),in
```

executed in 88ms, finished 15:31:41 2020-06-26

In [158]:

```
1 data.head()
```

executed in 36ms, finished 15:31:41 2020-06-26

Out[158]:

	발생지시도	사고유형_대분류	도로형태_대분류	당사자종별_1당_대분류	사상자분류	주야분류
0	0	차대사람	단일로	승용차	1	1
1	1	차대차	단일로	승용차	3	1
2	2	차량단독	단일로	승용차	1	0
3	3	차대차	교차로	승합차	2	0
4	4	차량단독	단일로	승용차	1	1

In [159]:

```
1 data.iloc[5:10]
```

executed in 34ms, finished 15:31:41 2020-06-26

Out[159]:

	발생지시도	사고유형_대분류	도로형태_대분류	당사자종별_1당_대분류	사상자분류	주야분류
5	4	차량단독	단일로	승용차	1	0
6	3	차량단독	단일로	승용차	2	0
7	4	차량단독	단일로	원동기장치자전거	1	1
8	2	차량단독	단일로	승용차	2	1
9	0	차대차	교차로	승합차	3	1

4.2.2 다른 컬럼 이제 바꾸기

In [160]:

```
1 lookup_LoadType_value = dict(zip(data.도로형태_대분류.unique(), range(len(data.도로형태_대분류
2 data['도로형태_대분류'].replace(list(lookup_LoadType_value.keys()),list(lookup_LoadType_value
```

executed in 76ms, finished 15:31:41 2020-06-26

In [161]:

```
1 lookup_AttackCarType_value = dict(zip(data.당사자종별_1당_대분류.unique(), range(len(data.당사
2 data['당사자종별_1당_대분류'].replace(list(lookup_AttackCarType_value.keys()),list(lookup_Atta
3
4 # 사고유형_대분류만 목표 변수니까 이름을 value로 하겠다.
5 lookup_AccidentType_name = dict(zip(range(len(data.사고유형_대분류.unique())), data.사고유형_
6 data['사고유형_대분류'].replace(list(lookup_AccidentType_name.values()),list(lookup_AccidentTy
```

executed in 168ms, finished 15:31:41 2020-06-26

In [162]:

```
1 data.head()
```

executed in 26ms, finished 15:31:41 2020-06-26

Out[162]:

	발생지시도	사고유형_대분류	도로형태_대분류	당사자종별_1당_대분류	사상자분류	주야분류
0	0	0	0	0	1	1
1	1	1	0	0	3	1
2	2	2	0	0	1	0
3	3	1	1	1	2	0
4	4	2	0	0	1	1

In [163]:

```
1 lookup_DayType_value = {'주간': 0, '야간':1}
2 lookup_DayType_value
```

executed in 11ms, finished 15:31:41 2020-06-26

Out[163]:

```
{'주간': 0, '야간': 1}
```

In [164]:

```
1 def lookup_CasualtiesType_value(num):
2     if num >=3 :
3         return 3
4     else :
5         return num
```

executed in 8ms, finished 15:31:41 2020-06-26

In [165]:

```
1 lookup_CasualtiesType_value(65)
```

executed in 11ms, finished 15:31:41 2020-06-26

Out[165]:

```
3
```

4.2.3 만들어진 lookup dictionary 확인하기

In [166]:

```
# 입력변수 관련 dictionary 확인.
print(lookup_Area_value)
print()
print(lookup_LoadType_value)
print()
print(lookup_AttackCarType_value)
print()
# lookup_CasualtiesType_value 함수 확인.
print(lookup_DayType_value)
print()
# 목표 변수 관련 dictionary 확인.
print(lookup_AccidentType_name)

```

executed in 15ms, finished 15:31:41 2020-06-26

```
{'서울': 0, '전북': 1, '충남': 2, '경남': 3, '경북': 4, '전남': 5, '충북': 6, '부
산': 7, '경기': 8, '인천': 9, '광주': 10, '울산': 11, '강원': 12, '대구': 13, '제
주': 14, '대전': 15, '세종': 16}

{'단일로': 0, '교차로': 1, '기타/불명': 2, ' 횡단 후진등 금지위반(부당한 회전)': 3,
'어린이보호위반': 4, ' 최저속도': 5, '보행자통행방해행위)': 6, '유턴': 7, '기타': 8,
'불명': 9, '주차장': 10}

{'승용차': 0, '승합차': 1, '원동기장치자전거': 2, '이륜차': 3, '화물차': 4, '농기
계': 5, '특수차': 6, '자전거': 7, '교차로부근': 8, '건설기계': 9, '단일로': 10, '교
차로내': 11, '기타단일로': 12, '불명': 13, '횡단보도부근': 14, '횡단보도상': 15, '교
차로': 16, '기타/불명': 17, '고가도로위': 18, '교량위': 19, '사륜오토바이(ATV)': 20,
'기타': 21, '개인형이동수단(PM)': 22, '교차로횡단보도내': 23, '지하차도(도로)내': 2
4}

{'주간': 0, '야간': 1}

{0: '차대사람', 1: '차대차', 2: '차량단독'}
```

5 Predictive Modeling

저는 입력 변수(독립변수)를 당사자종별_1당_대분류, 도로형태_대분류, 발생지시도, 사상자분류, 주야분류 로 하였습니다.

그리고 출력변수(종속변수)를

사고유형_대분류로 하였습니다.

그래서 사고유형_대분류를 추정할 수 있는 모델을 만들었습니다.

In [167]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score

```

executed in 11ms, finished 15:31:42 2020-06-26

In [168]:

```
data.head()
```

executed in 26ms, finished 15:31:42 2020-06-26

Out[168]:

	발생지시도	사고유형_대분류	도로형태_대분류	당사자종별_1당_대분류	사상자분류	주야분류
0	0	0	0	0	1	1
1	1	1	0	0	3	1
2	2	2	0	0	1	0
3	3	1	1	1	2	0
4	4	2	0	0	1	1

In [169]:

```
# X_fruits = fruits[['height', 'width', 'mass', 'color_score']]
# y_fruits = fruits['fruit_label']
X_data = data[data.columns.difference(['사고유형_대분류'])]
y_data = data[['사고유형_대분류']]
```

executed in 24ms, finished 15:31:42 2020-06-26

In [170]:

```
data.columns.difference(['사고유형_대분류'])
```

executed in 15ms, finished 15:31:42 2020-06-26

Out[170]:

```
Index(['당사자종별_1당_대분류', '도로형태_대분류', '발생지시도', '사상자분류', '주야
분류'], dtype='object')
```

In [171]:

```
1 X_data.head()
```

executed in 29ms, finished 15:31:42 2020-06-26

Out[171]:

	당사자종별_1당_대분류	도로형태_대분류	발생지시도	사상자분류	주야분류
0	0	0	0	1	1
1	0	0	1	3	1
2	0	0	2	1	0
3	1	1	3	2	0
4	0	0	4	1	1

In [172]:

```
1 y_data.head()
```

executed in 19ms, finished 15:31:42 2020-06-26

Out[172]:

	사고유형_대분류
0	0
1	1
2	2
3	1
4	2

5.1 데이터 나누기

In [173]:

```
1 X_train, X_test, y_train, y_test = train_test_split(X_data, y_data, random_state = 0)
```

executed in 26ms, finished 15:31:42 2020-06-26

5.2 학습하고 평가하기

multi-classification이라서 precision score와 recall score를 계산할 수 없었습니다.

In [174]:

```
models = []
models.append(('LR', LogisticRegression()))
models.append(('SVM', SVC()))
models.append(('DT', DecisionTreeClassifier()))
models.append(('RF', RandomForestClassifier()))

names = []
# precisionList = []
# recallList = []
clfTrainingScoreList=[]
clfTestScoreList=[]

for name, model in models :
    model.fit(X_train, y_train)
    clfTrainingScoreList.append(model.score(X_train, y_train))
    clfTestScoreList.append(model.score(X_test, y_test))
#    y_predicted = model.predict(X_test)
    names.append(name)
#     precisionList.append(precision_score(y_test, y_predicted))
#     recallList.append(recall_score(y_test, y_predicted))
#  멀티 클래시피케이션이라

for name, trainingScore, testScore in zip(names, clfTrainingScoreList, clfTestScoreList):
    #print(names[i], 'precision : ', '{:.2f}'.format(precisionList[i]), 'recall : ', '{:.2f}'
    print(name, 'trainingScore : ', '{:.2f}'.format(trainingScore), 'testScore : ', '{:.2f}'.
```

executed in 1m 57.1s, finished 15:33:39 2020-06-26

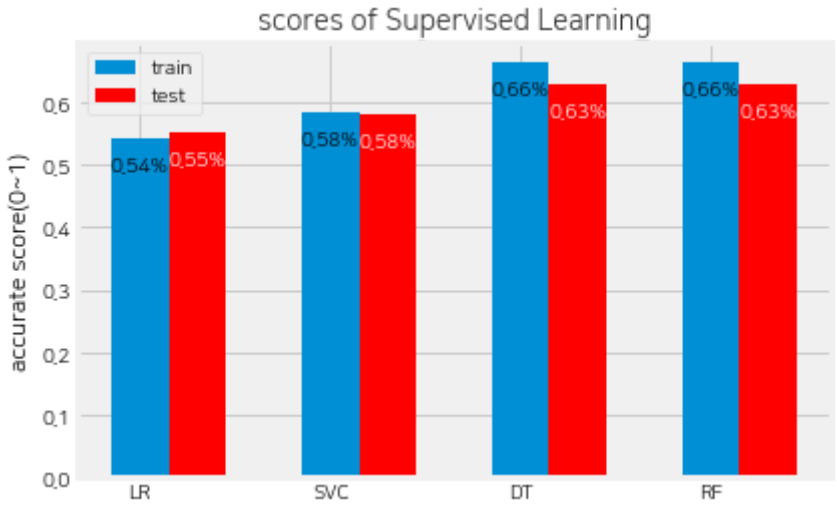
```
LR trainingScore :  0.54 testScore :  0.55
SVM trainingScore :  0.58 testScore :  0.58
DT trainingScore :  0.66 testScore :  0.63
RF trainingScore :  0.66 testScore :  0.63
```

5.3 그래프로 표현하기

In [175]:

```
plt.figure()
xvals = range(len(clfTrainingScoreList))
bar1 = plt.bar(xvals, clfTrainingScoreList, width = 0.3)
new_xvals = []
for item in xvals:
    new_xvals.append(item + 0.3)
bar2 = plt.bar(new_xvals, clfTestScoreList, width = 0.3 , color = 'red')
nameList = ['LR', 'SVC', 'DT', 'RF']
pos = np.arange(len(nameList))
plt.xticks(pos, nameList)
plt.ylabel('accurate score(0~1)')
plt.title('scores of Supervised Learning', alpha=0.8)
plt.legend(['train', 'test'])
for bar in (bar1):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height() -0.05,
        '%.2f' % float(bar.get_height()) + "%",
        horizontalalignment='center',
        color='black',
        #weight='bold',
        #fontsize=11
)
for bar in (bar2):
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        bar.get_height() -0.05,
        '%.2f' % float(bar.get_height()) + "%",
        horizontalalignment='center',
        color='white',
        #weight='bold',
        #fontsize=11
)
```

executed in 708ms, finished 15:33:39 2020-06-26



어떤 모델에서 테스트셋 구분에 가장 좋은 성능을 보이는지 보았더니 렌덤 포레스트와 의사결정 트리로 밝혀졌습니다.

5.4 예측해보기

그렇다면 사상자수가 3명이상 인 사고,

주간에 서울에서 단일로에서 승용차 사고로 사상자수가 4명이 발생했을때, 그 사고의 사고유형 대분류를 예측해 보겠습니다.

In [176]:

```
1 X_data.head()
```

executed in 19ms, finished 15:33:39 2020-06-26

Out[176]:

	당사자종별_1당_대분류	도로형태_대분류	발생지시도	사상자분류	주야분류
0	0	0	0	1	1
1	0	0	1	3	1
2	0	0	2	1	0
3	1	1	3	2	0
4	0	0	4	1	1

In [177]:

```
1 accidentType_prediction = models[3][1].predict([[lookup_AttackCarType_value['승용차'],
2                                                  lookup_LoadType_value['단일로'],
3                                                  lookup_Area_value['서울'],
4                                                  lookup_CasualtiesType_value(4),
5                                                 lookup_DayType_value['주간']]])
```

executed in 41ms, finished 15:33:40 2020-06-26

In [178]:

```
1 lookup_AccidentType_name[accidentType_prediction[0]]
```

executed in 10ms, finished 15:33:40 2020-06-26

Out[178]:

'차대차'

아, 차대차 사고일 것이구나 예측할 수 있습니다.

끝까지 읽어주셔서 감사합니다.!