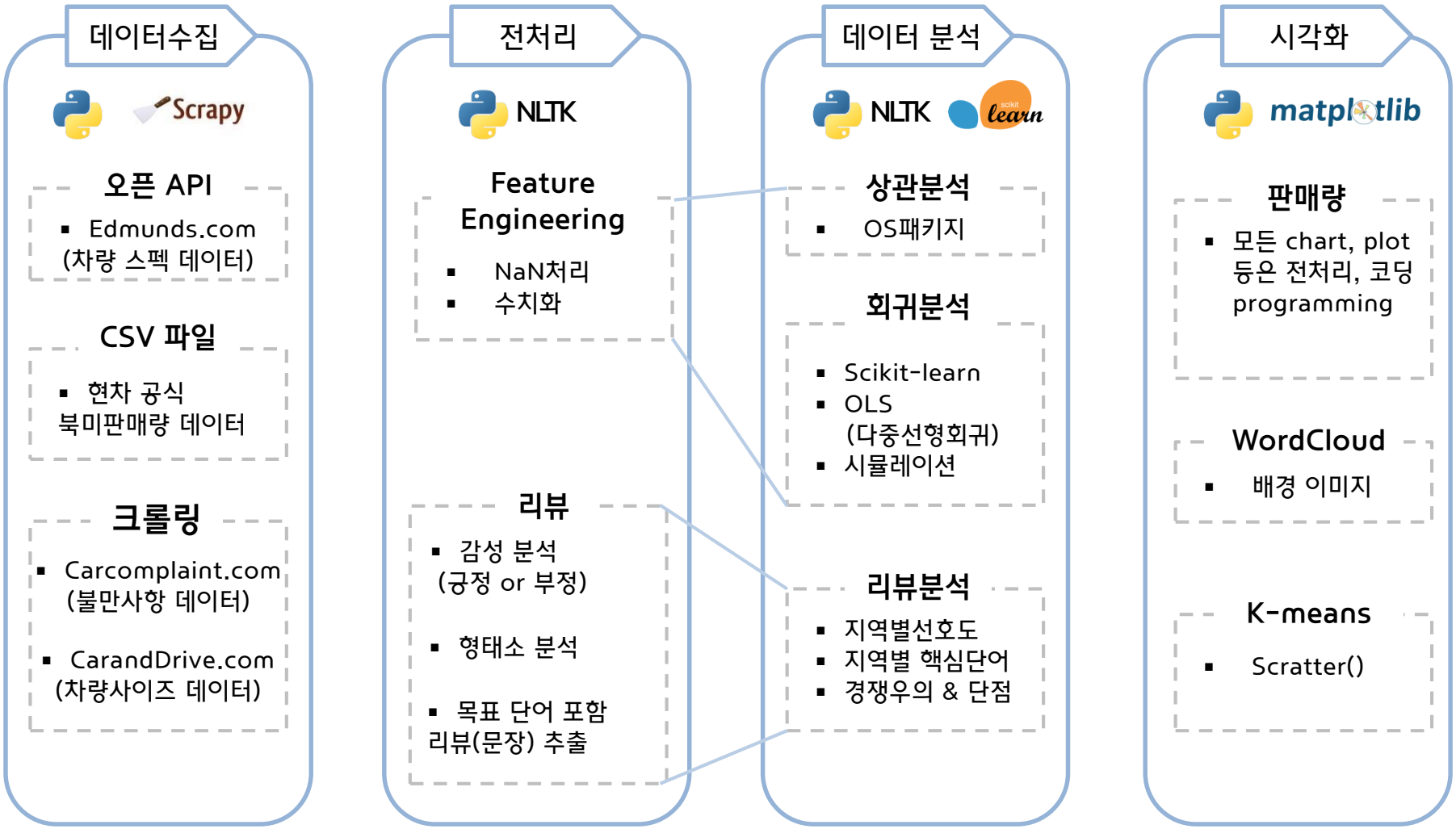
* 본 문서에는 기업과 관련된 정보를 포함할 수 없습니다.

Maxcruz 북미 상품성 개선 전략

2017-08-23

Data Analyzing Process



- 분석 시작 전 “큰 그림 " 을 그리는 것이 중요하다.
- 특히, **“종속변수“ (예측치)** 를 설정하고 이를 맞출 수 있는 독립 변수들, 머신러닝 or 분석 기술들을 간략하게 선정하고 시작하는 것이 가장 중요하다.
- 결국, **“수치적”**으로 보여주는 것이 “데이터 분석 " 의 목적이다.

Skills

- 종속변수를 “판매량”으로 설정하고 상품성 상승 -> 판매량 상승 을 목적으로 분석을 진행한다.
- 독립변수는 크게 “리뷰 and 평점” 과 “스펙 and 사이즈” 로 선정한다.
- 소비자들에게 평이 안좋은 차량 선정 -> 경쟁차량 or Segment차량 비교 분석 -> 결과 시각화

리뷰분석

- 수치화 할 수 있는 부분을 정량화 하여 평점이 가장 낮은 차량을 선정
- 정성적(리뷰)요소는 " 감성 분석”을 통해 단점인 문장을 추출하고 " 형태소 분석“(명사-전처리)를 통해 단어 조합을 만든다.

리뷰 분석기 MODEL

INPUT

Raw Review

“I think Maxcruz`s cargospace is **little smaller** than other competiton cars”

PROCESSING

부정 문장 → Cargosapace (명사) Smaller (비교)

OUTPUT

Word combination

Maxcruz -> “Smaller cargospace”

스펙분석

- 크기, 가격을 기준으로 군집화(Kmeans) : 소비자 차량구매의 기본 준거
- 경쟁 차량 기준 스펙 상세 비교 -> 개선 point 생각
- point 에 유의미한 변수 “회귀분석(OLS)”를 통해 도출
- 회귀분석을 통한 “판매량 " 예측 모델링

전장, 전고, 마력, 연비, · · ·

OLS 판매량 예측 MODEL

유의미한 변수 : 전장, 전고 · · ·

판매량 = $Ax_1 + Bx_2 + Cx_3 - Dx_4 + \alpha$

< 목차 >

[Modeling] 데이터 수집 to 크롤링

In [3]:
```
def get_post_link(url): # url은 hyundai/santafe/2017 즉, make,model,year을 변경해줘야한다.
    target_url = url
    res = urllib.request.urlopen(target_url)

    html = res.read()
    soup = BeautifulSoup(html, 'html.parser')

    root_path = soup.find('div', id='graph').find('ul')
    second_list = root_path.find_all('li')

    third_path =[]
    # 최종 path를 담은 list

    for path in second_list:
        second_path = target_url + path.find('a').get('href')
        res = urllib.request.urlopen(second_path)
        html = res.read()
        soup = BeautifulSoup(html, 'html.parser')
        third_list = soup.find('div', id='graph').find('ul').find_all('li')

        for path in third_list:
            third_path.append(second_path + path.find('a').get('href'))
            # 문자열인 path를 연결해서 최종 third_list path를 만들어낸다.

    return third_path
```

- 크롤링을 할 사이트의 url을 읽고 원하는 정보가 포함된 url을 정해주는 code

In [4]:
```
def get_post_review(link): # 반복문을 통해 모든 link를 받아들인다.

    Big_category = link.split('/')[-2]
    Small_category = link.split('/')[-1].split('.')[0]
    # category는 link에 포함되어 있는 내용이다.

    url = link
    res = urllib.request.urlopen(url)
    html = res.read()
    soup = BeautifulSoup(html, 'html.parser')


    complaint_path = soup.find('div', id='pcomments').find_all('div', class_='complaint')
    # 각 link당 homepage에 보여지는 각각의 'Complain 항목' 들

    review_head =  [comment_path.find('div', class_='cheader')
                    for comment_path in complaint_path ]

    # 각각의 'Complain'마다 date / year / title / sub_title 을 포함하는 구역
```

...

```
    return{
        'title' : title,
        'sub_title' : sub_title,
        'year' : year,
        'date' : date,
        'country' : country,
        'city' : city,
        'text' : text,
        'B_category' : B_category,
        'S_category' : S_category
    }
```

- Get_post_link(url)에서 읽은 URL을 따라 원하는 data만 전처리, 수집하는 code

Code : Python_codes/Data_collecting/Main_analyzing/Data_Crawling_Complain.ipynb

[Modeling] 현황 분석 전처리 & 시각화 알고리즘

In [99]:
```
dummy_table = pd.DataFrame(np.arange(1).reshape(1,1))

## change
label_list = ['PUP-C','PUP-D']

for label in label_list:
    new_table = pd.DataFrame(Pick_pivot[Pick_pivot['Index']==label].sort_values('Year')['Amount'])
    new_table.index = [ x for x in range(len(new_table))]
    new_table.columns = [label]
    hap_table = pd.concat([dummy_table, new_table], axis=1)
    dummy_table = hap_table

hap_table = hap_table.drop([0],axis=1)

val_index = []
for j in hap_table.columns:
    for i in hap_table.index:
        label = str(round(hap_table.loc[i][j],1)) + "%"
        val_index.append(label)

fig, ax = plt.subplots(1,1)

fig.set_size_inches(15,10)

dates = np.arange(5)
labels = label_list
```

	GLOBAL_SEGMENT	Y2012	Y2013	Y2014	Y2015	Y2016
0	SUV-B	1.076710	1.492728	1.647823	4.476272	5.956089
1	SUV-C	43.616571	45.094528	48.377932	47.426271	45.962639
2	SUV-D	35.188496	33.625746	30.646859	30.609713	29.877648
3	SUV-E	20.118224	19.786998	19.327387	17.487744	18.203624

- 각각의 barplot에 labeling을 하는 code

- Text()를 사용하여 x축에 따른 세부 barplot에 labeling
- 범례, 축, 타이틀, 세부 color 등 지정

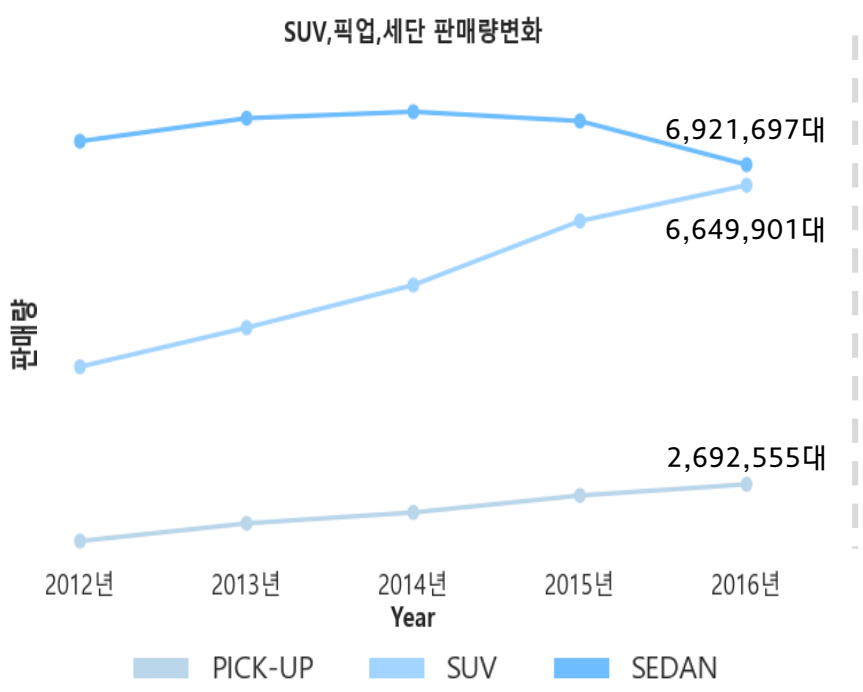
```
## change
current_palette = sns.color_palette("Blues")
test = [current_palette[1], '#a1d4ff', '#6dbdff']
colors= ['#a1d4ff','#84c7ff','#77c1ff']
margin_bottom = np.zeros(5)

for index, label in enumerate(labels):
    values = Pick_pivot[Pick_pivot['Index']==label].sort_values('Year')['Amount'].apply(float)
    ax.bar(dates, values,
           align='center', width=0.5, label=label, color=test[index], bottom=margin_bottom)
    margin_bottom += values

patches = ax.patches
for label, rect in zip(val_index, patches):
    width = rect.get_width()
    if width >0:
        x = rect.get_x()
        y = rect.get_y()
        height = rect.get_height()
        ax.text(x+width/2., y+height/2., label, ha='center', va='center', fontsize=20, weight='bold')

l = ax.legend(labels,loc='upper center', bbox_to_anchor=(0.5, -0.09),
          fancybox=True, shadow=True, ncol=6, prop={'size':20})

l.get_texts()[0].set_text('중형 픽업트럭')
l.get_texts()[1].set_text('풀사이즈 픽업트럭')

plt.xlabel('Year', fontsize=20, weight='bold')
plt.ylabel('점유율', fontsize=20, weight='bold')
plt.tick_params(axis='both', which='major', labelsize=20)

ax.set_xticklabels(['0','2012년','2013년','2014년','2015년','2016년'])

ax.set_facecolor('white')
plt.title('픽업트럭 Segment 점유율 변화', fontsize=20, weight='bold')
plt.savefig('Pick-up 점유율 변화.png', transparent=True)
```

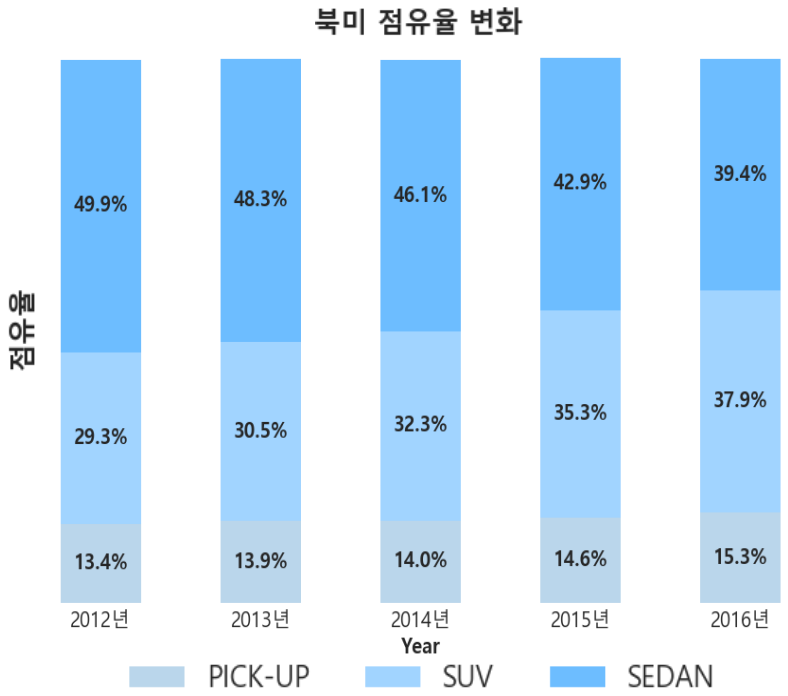
Code : Python_codes/US_market_analyzing/시장현황.ipynb

▪ SUV 시장의 성장

- SUV Market Share은 17년도 Sedan시장을 앞지를 것이라 예측된다.

북미 판매량 및 주요 Segment 성장 추이





자료 : 현대자동차 공식 북미 판매량 DataSet
주 : Segment 기준은 현대자동차 공식 판매량 DataSet의 GLOBAL_SEGMENT

[생략] 시장현황 분석 시각화

- 북미 판매량 변화 추이, Segment별 추이

- 현대자동차 SUV 차량 점유율 변화

- 경쟁업체 비교

- 등등…

[Modeling] 군집화 알고리즘


```
feature = ['pass_capa', 'f_head', 'f_leg','f_shoulder', 'f_hip', 's_head', 's_leg', 's_shoulder', 's_hip',
        'wheelbase', 'length', 'width', 'height','track_width_front', 'track_width_rear',
            'baseMSRP','full_size','cargo_space','pass_vol']
cluster_value = data[feature].values


kmeans = KMeans(n_clusters=3, random_state=1)
kmeans.fit(cluster_value)

labels = ['MidLarge_SUV','Midsize_SUV','Largesize_SUV' ]
plot_x = pd.DataFrame(cluster_value)[15]
plot_y = pd.DataFrame(cluster_value)[16]


label_color = {0:'red',1:'green',2:'blue'}
current_palette = sns.color_palette("Blues")
test = [current_palette[1], '#a1d4ff', '#6dbdff']
colors =[current_palette[1],'#359cfb','blue']

fig, ax = plt.subplots(1, figsize=(15,8))
mglearn.discrete_scatter(cluster_value[:, 15], cluster_value[:, 16], kmeans.labels_, markers='o',c=colors, s=13)




plt.xlabel('Price', fontsize=15, weight='bold')
plt.ylabel('Exterior+Interior Size', fontsize=15, weight='bold')
plt.tick_params(axis='both', which='major', labelsize=15)

ax.legend(labels,loc='upper center',bbox_to_anchor=(0.5, -0.08),
          fancybox=True, shadow=True, ncol=4, prop={'size':15})

ax.annotate('Maxcruz', xy=(cluster_value[30][15], cluster_value[30][16]), xytext=(cluster_value[30][15]-13000, cluster_value[30][16]+15),
            arrowprops=dict(facecolor='black', shrink=0.05), fontsize=20, color='black')

ax.annotate('Santa Fe', xy=(cluster_value[31][15], cluster_value[31][16]), xytext=(cluster_value[31][15]-8500, cluster_value[31][16]-15),
            arrowprops=dict(facecolor='black', shrink=0.02), fontsize=20, color='black')



plt.title('첫번째 Segment', fontsize=17, weight='bold')

plt.savefig('첫번째 Segment.png')

plt.show()
```

```
from sklearn.cluster import KMeans
import mglearn
```

- Sklearn의 Kmeans 모듈을 사용
- Discreate_scatter 모듈을 사용하여 군집 시각화
- Pyplot 내부 tool 사용

Code : Python_codes/Segment_clustering/Main_analyzing/Segment-clustering.ipynb

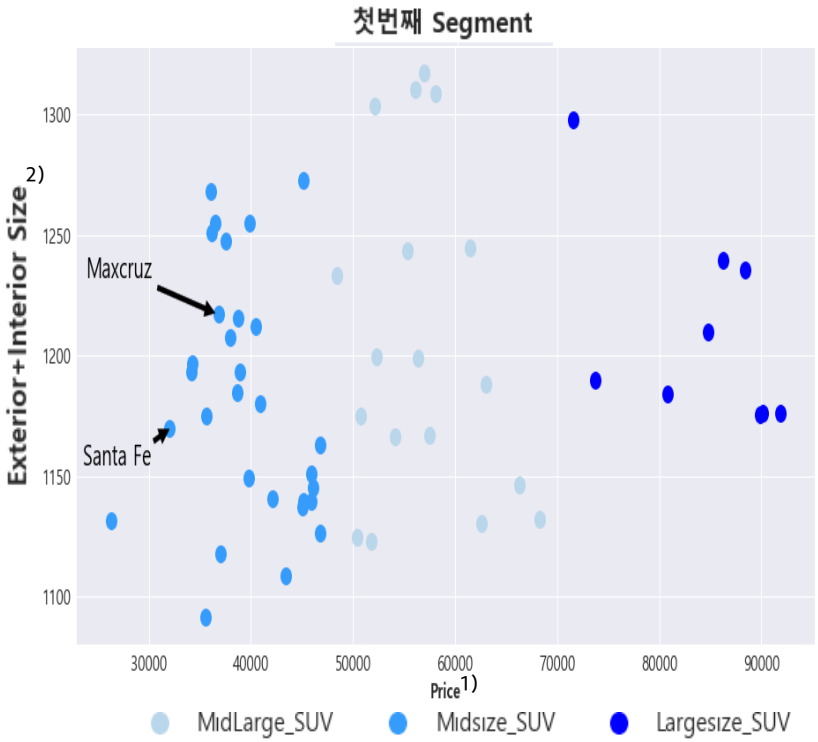
2. 경쟁차량 선정

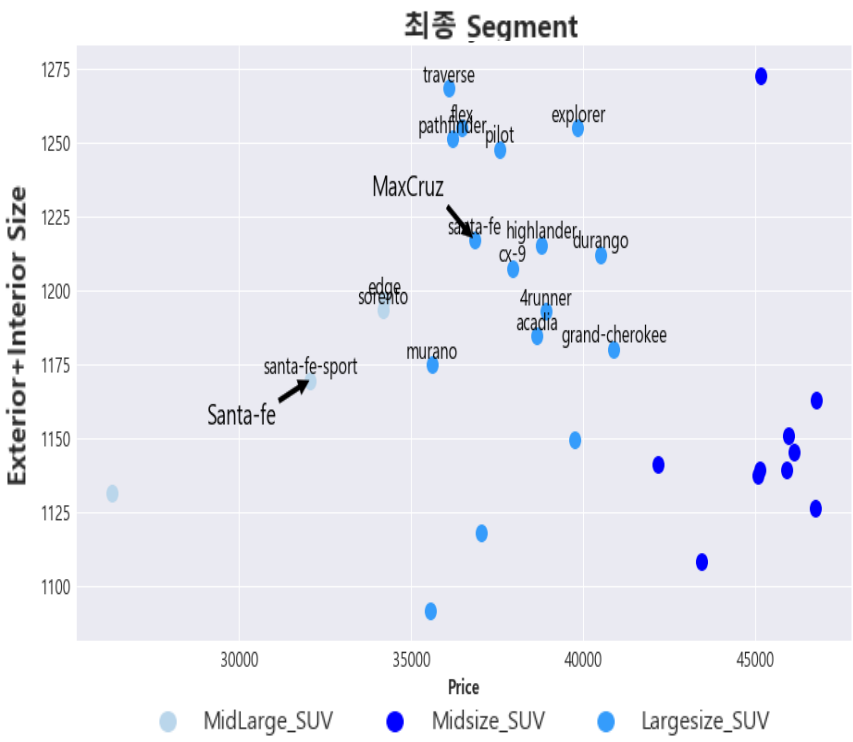
Santa Fe와 Maxcruz 군집과 경쟁 차량 확인

- **최종 군집을 통해 확인 가능한 경쟁 차량**

  - 1차 : Honda : Pilot / Toyota : Highlander, 4runner / Nissan : Pathfinder, Murano / Mazda : CX-9 / Ford : Edge,Flex, Explorer / GMC : Acadia / Chevrolet : Traverse / Kia :Sorento / Jeep : Grand-Cherokee, / Dodge : Durango (14개)

Kmeans 군집생성





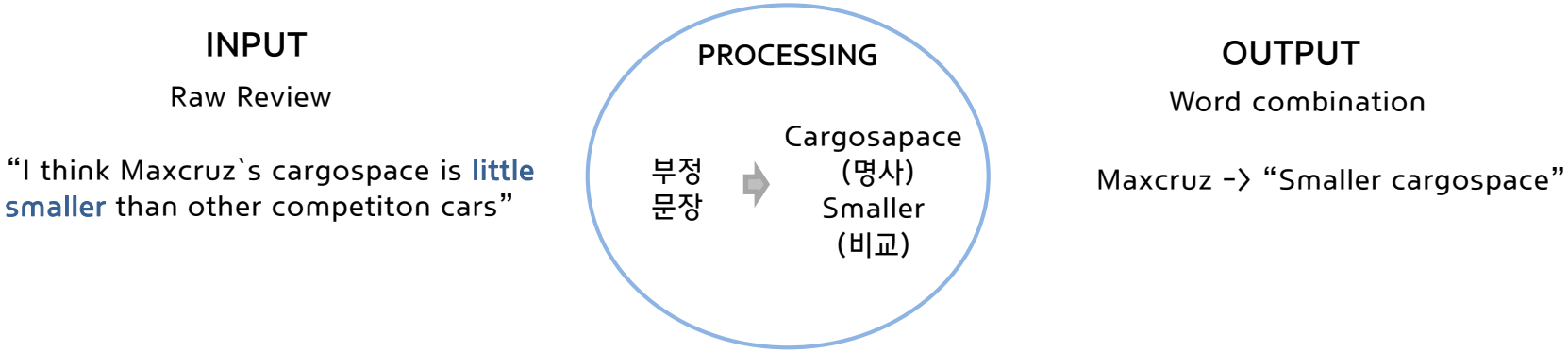
자료 : Edmunds.com, CarandDrive.com 크롤링 , model 공식 Homepage Spec(Dimension) Data

주 : 1) MSRP(권장 소비자 가격) 2) 전장, 전폭, 휠베이스 등 (6개) + 앞· 뒤 좌석 치수(머리, 다리, 어깨, 힙), 화물· 승객 용량 등 (12개)

HYUNDAI

[Modeling] 리뷰 분석기

리뷰 분석기 MODEL



- NLTK 패키지의 VADER 모듈을 사용
  1. 특정 자동차 리뷰들을 수집하여 list형태로 변환한다.
  2. 각 리뷰를 긍정 or 부정 으로 나눈다.
  3. **'긍정 리뷰'**에서 많이 도출되는 단어 와 **'부정 리뷰'** 에서 많이 도출되는 단어를 비교한다.
  4. 경쟁차량들의 리뷰도 동일한 방법으로 비교한다.
  5. 개선 Point Insight를 도출한다.

[Insight 결과] Maxcruz는 경쟁차량과 비교해서 공간(외부, 내부 사이즈, 편안함 등)의 '불만 단어'가 자주 출현한다.

Code : Python_codes/Review_analyzing/Main_analyzing/Review Modeling.ipynb

[생략] 경쟁 차량과 1차원적 비교(리뷰 살피기, 사이즈, 스펙 비교 등..)

- 경쟁차량 선정 기준(리뷰, 공식홈페이지비교)

- 최종 선정된 경쟁 차량 Pilot, Pathfinder, Highlander 와 Maxcruz 스펙 상세비교

- Maxcruz 리뷰 평점 42개 항목의 점수를 “평균 " 내어 경쟁차량과 비교 후 Maxcruz 단점 도출

- Maxcruz 단점 변수(화물용량)개선을 ‘Point’로 생각하여 이를 종속변수로 하여 상관분석 실행

- 상관분석 결과 ‘전고’, ‘전장’ 독립변수를 중요 개선 변수로 최종 선택

[Modeling] 판매량 예측 회귀 Modeling Process - OLS

In [323]:
```
feature_x = ['cargo_space','pass_vol','pass_capa','length','wheelbase','baseMSRP']
feature_y = 'Y2016'

train_x = data[feature_x]
train_y = data[feature_y]

train_x2 = statsmodels.tools.tools.add_constant(train_x)
train_y2 = pd.DataFrame(train_y)
result = sm.OLS(train_y2, train_x).fit()
result.summary()
```

Out [323]:

OLS Regression Results

Dep. Variable:	Y2016	**R-squared:**	0.655
Model:	OLS	**Adj. R-squared:**	0.615
Method:	Least Squares	**F-statistic:**	16.14
Date:	Sun, 13 Aug 2017	**Prob (F-statistic):**	2.65e-10
Time:	19:34:13	**Log-Likelihood:**	-689.99
No. Observations:	57	**AIC:**	1392.
Df Residuals:	51	**BIC:**	1404.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
cargo_space	602.6045	434.408	1.387	0.171	-269.506 1474.715
pass_vol	564.5462	401.127	1.407	0.165	-240.751 1369.843
pass_capa	-1.436e+04	8923.724	-1.609	0.114	-3.23e+04 3560.022
length	3635.8515	1848.450	1.967	0.055	-75.066 7346.769
wheelbase	-5549.0753	2896.237	-1.916	0.061	-1.14e+04 265.363
baseMSRP	-1.0995	0.407	-2.703	0.009	-1.916 -0.283

Omnibus:	27.408	**Durbin-Watson:**	2.081
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	51.246
Skew:	1.559	**Prob(JB):**	7.45e-12
Kurtosis:	6.443	**Cond. No.**	8.15e+04

- 회귀 식 도출 (Page)
- R-squared 가 뛰어나게 높진 않지만 독립변수들의 영향력을 살필 수 있다.

Code : Python_codes/Compare_competition_cars/Main_analyzing/판매량 예측 회귀분석 모델.ipynb

[Modeling] 판매량 예측 회귀 Modeling Process2 - 시뮬레이션

In [319]:
```
dummy2 = pd.DataFrame(np.arange(1).reshape(1,1))
for x in tqdm(range(1000)):
    y_list = []
    dummy = pd.DataFrame(np.arange(1).reshape(1,1))
    for x in range(100):
        x1= random.uniform(193.1, 198.5)
        x2= random.choice([7,8])
        x3= random.uniform(156, 173.8)
        x4= random.uniform(79.8, 83.9)
        x5= random.uniform(109.8, 114.2)
        x6= 30800



        y= 3635.8*x1-0.001*x2+714.6*x3+560.4*x4-5549*x5-1.09*x6-190000
        table = pd.DataFrame([y,x1,x2,x3,x4,x5,x6]).T
        table.columns = ('판매량','length','pass_capa','pass_vol','cargo_space','wheelbase','baseMSRP')
        result_t = pd.concat([dummy,table])
        dummy = result_t

    result_t.index = [ x for x in range(len(result_t))]
    result_t = result_t.drop(0)
    result_t = result_t.drop([0],axis=1)
    first_table = result_t.sort_values('판매량',ascending=False).head(1)

    final_t = pd.concat([dummy2, first_table])
    dummy2 = final_t

final_t.index = [ x for x in range(len(final_t))]
final_t = final_t.drop(0)
final_t = final_t.drop([0],axis=1)
```

```
  0%|                                                                                | 0/1000 [00:00<?, ?it/s]C:₩Users₩kb910₩Anacon
da3₩lib₩site-packages₩pandas₩indexes₩range.py:432: RuntimeWarning: '<' not supported between instances of 'int' and 'str', sort order is und
efined for incomparable objects
  return self._int64index.union(other)
100%|████████████████████████████████████████████████████████████████████| 1000/1000 [04:22<00:00,  3.84it/s]
```

In [320]:
```
final_t.mean()
```

Out[320]:
```
판매량           52606.151055
length         197.865012
pass_capa        7.506000
pass_vol       170.832497
cargo_space     82.129032
wheelbase      110.167862
baseMSRP     30800.000000
dtype: float64
```

- 도출된 회귀식에 시뮬레이션을 적용
- 난수 생성을 통해 총 100*1000번의 시뮬레이션 진행
- 각 시뮬레이션의 평균을 도출

Code : Python_codes/Compare_competition_cars/Main_analyzing/판매량 예측 회귀분석 모델.ipynb

HYUNDAI

[생략] 회귀식을 통한 최종 차량 스펙 결정, 최종 결론

- 상관분석을 통한 판매량 예측 회귀식 도출(OLS 모델)

- 시뮬레이션 에 따른 최종 차량 수치 결정

3. 개선점 도출

지역별 개선 Point 도출

- SUV 60종의 리뷰 약 25,000건을 “크롤링”을 통해 수집
- 25,000건의 리뷰를 각 지역별로 분할
- 가장 리뷰 수 가 적은 서부(2605)개에 맞춰 random으로 각 지역별 2605개의 리뷰 추출
- 각 지역별 리뷰를 문장으로 나누고 원하는 “단어 사전(명사)”만 전처리 후 빈도 비율 “수치화”
- 최종 수치(Score) = (각 지역별 단어 빈도 – 그 단어의 전체 지역 평균 빈도) 를 도출
- Score : (-), (+)의 값이 큰 단어, ‘0’인 단어를 그 지역의 유의미한 단어로 판단
- 유의미한 단어에 맞는 각 지역별 Maxcruz의 핵심 “기능(열선, 스마트 컨트롤 등등)”을 적용[생략]

Out[70]:

	Central	Central_score	West	West_score	East	East_score	South	South_score
0	pedal	-0.113943	freeway	-0.142624	highway	-0.141672	safety	-0.124362
1	stop	-0.105823	park	-0.069976	brakes	-0.057373	mph	-0.096856
2	mph	-0.096856	fuel	-0.038732	reverse	-0.030786	accident	-0.081620
3	gas	-0.093141	gas pedal	-0.037969	stop	-0.023494	stop	-0.071344
⋮								
29	mph warning	0.000000	gas gas	0.000000	truck	0.000000	water	0.000000
30	brakes stop	0.000000	smoke	0.000000	visibility	0.000000	exhaust	0.000000
31	police	0.000000	stop stop	0.000000	accelerator pedal	0.000000	emergency	0.000000
			stop sign	0.000000			tire tire	0.000000
			indicator	0.000000				
⋮								
35	engine	0.016368	air	0.079698	way	0.064501	gas pedal	0.031644
36	interstate	0.034794	warning	0.092001	fuel	0.072589	engine engine	0.050334
37	accident	0.041379	highway	0.138631	gas	0.095093	pedal	0.053604
38	highway	0.045857	stop	0.200661	traffic	0.103880	park	0.080263
39	brakes	0.077530	mph	0.202789	safety	0.146497	freeway	0.142624

Code : Python_codes/Compare_competition_cars/Main_analyzing/지역별 리뷰비교.ipynb

[생략] 최종 결론

- 기대효과

- 개선 방향성