빅데이터 분석 미니프로젝트 최종 보고서

201710791 류지혜

1.문제

* 문제 정의: 멜론 차트의 매월 순위 변동추이를 보고 사람들이 몇 월에 가장 새로운 노래를 많이 찾아 듣는지 분석하여 미래의 순위 변동추이를 예측한다.

* 문제 기술

WHAT 대중들이 새로운 음악을 많이 찾아 듣는 시기가 있을까?

WHO 음반제작사가 음원을 발매할 시기를 결정할 수 있는가?

WHY 음원의 수익이 대부분 스트리밍에서 발생하는 음반시장의 현황에서 음원사이트 중 가장 점유율이 가장 높은 사이트 '멜론'에서의 순위 변동 추이를 보고 미래의 순위 변화를 예측할 수 있다면 음반제작사에는 경쟁사들의 발매일과 예측값을 지표로 어느 시기에 음반을 발매할지 계획하는 것이 한층 수월해 질 것이다.

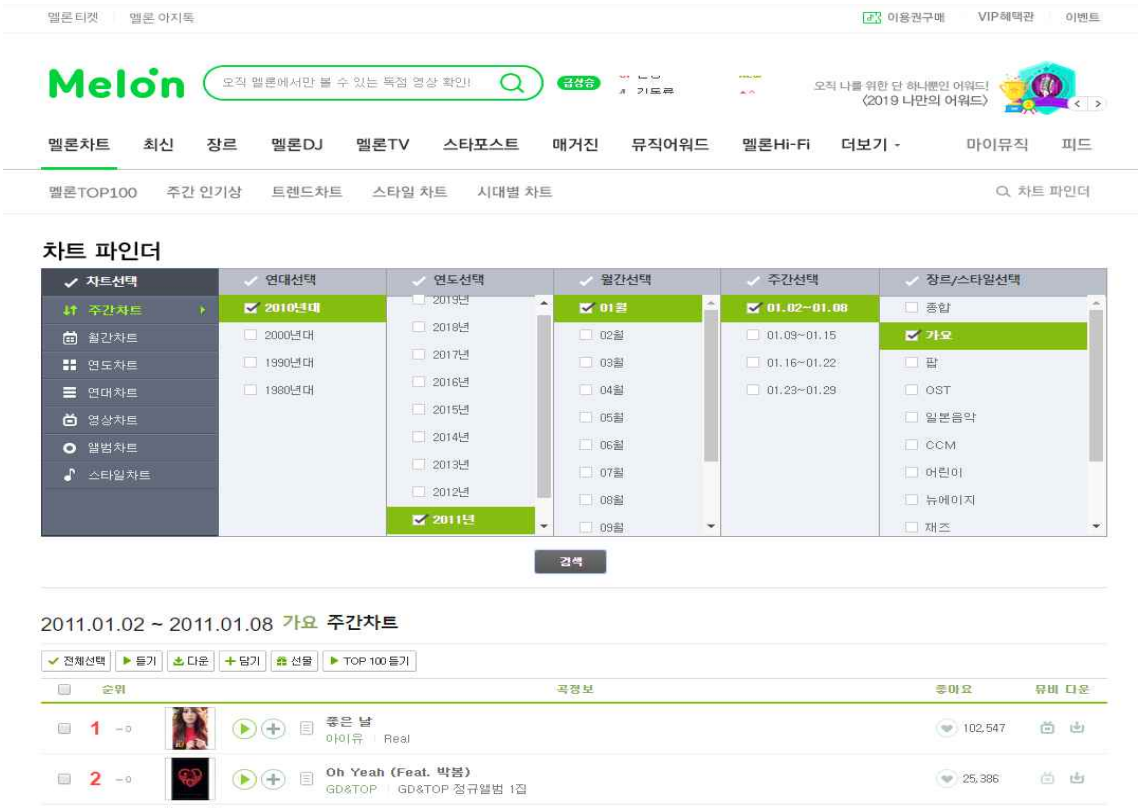
HOW

1. 멜론의 10년치 주간 차트에서 100위까지 노래의 정보를 크롤링 방식으로 수집
->(연도,월,주,순위,곡명,가수명,변동된 순위, 상태(상승/하락))
2. 수집된 빅데이터를 Data Frame으로 만들어 변동된 순위에 대한 변동지수를 설정하여 새로운 열을 추가한다.
3. 몇 월에 가장 변동이 심한지 확인할 수 있는 월별 변동지수의 총합에 대한 Data Frame과 연도별로 월을 피벗으로 하여 변동지수의 합에 대한 Data Frame을 생성한 후 그래프로 시각화한다.
4. 회귀분석을 위해 매 달에 대한 변동지수 합을 연도별로 리스트에 저장하여, 1월부터 12월까지 연도별 변동지수를 그래프로 나타낸 후, 매 월에 대한 회귀분석을 실시한다.
5. 회귀분석으로 매월 B0,B1값을 구했으면 정확도를 위해 Rsquared값을 구하고, 미래의 년도와 월을 입력하면 순위변동추이를 출력해주는 함수를 만들어 예측 시스템 설계완료한다.

2. 문제 해결 과정

1) 빅데이터 수집 : 크롤링

멜론의 차트파인더에서 10년간의 모든 주간차트에서 원하는 데이터를 추출해내기 위해 크롤링이 필요했습니다. Python3 환경에서 BeautifulSoup, Selenium을 이용하여 크롤링 하였고 전체적인 방법과 틀은 구글링을 참조했지만 제가 원하는 세부 정보를 추출하고, 여러 가지 예외 상황들을 처리하고 주간 차트를 선택하는 코드 등은 대부분 제가 작성하였습니다. 또한 올바른 코드를 작성하고 나서도 멜론 서버의 문제점 때문에 사이트에 접속할 수 없는 문제가 자주 발생하여 집에 있는 두 대의 컴퓨터를 이용하여 분산적으로 데이터를 크롤링 하였습니다. 이는 for문의 인자만 바꿔주면 되었기 때문에 간단했습니다.



멜론 차트파인더의 첫 화면입니다. 보시다시피 차트>연대>연도>월>주간>장르 순으로 선택할 수 있습니다. 저는 주간차트를 선택하고, 연대는 2010년대로 고정, for문을 통하여 연도, 월간,주간을 선택하는 3중첩문을 사용하도록 했고, 가요차트(2014년 이후로는 국내종합) 을 선택하여 정보를 크롤링했습니다.

아래로는 크롤링을 위해 제가 작성한 코드입니다.

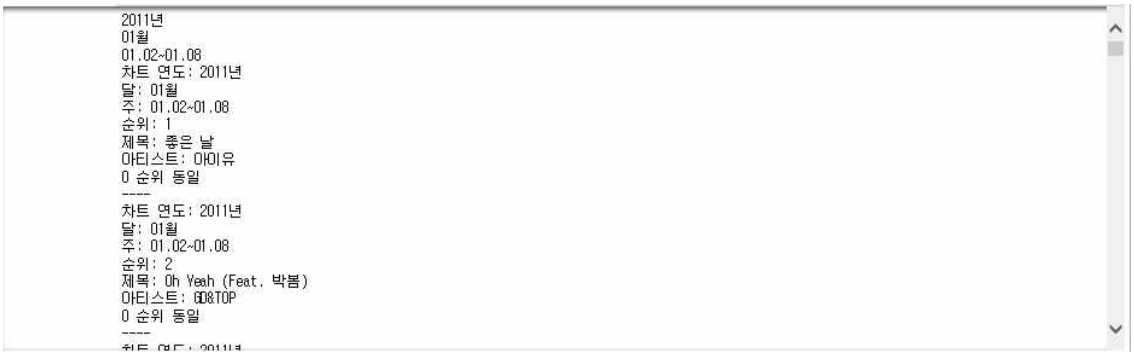
```
header = {'User-Agent': ''}
d = webdriver.Chrome('./Downloads/chromedriver')
d.implicitly_wait(3)
d.get('http://www.melon.com/chart/index.htm')
d.get("http://www.melon.com/chart/search/index.htm")
d.find_element_by_xpath('//*[@id="d_chart_search"]/div/h4[1]/a').click() #멜론의 차트파인더에서 주간차트 선택

req = requests.get('http://www.melon.com/chart/search/index.htm')
html = req.text
soup = BeautifulSoup(html, "html.parser")

#2010년대만 선택 3번째 div seotor의 인덱스가 연대-연도-월-주간-차트 순을 뜻함
age_xpath = '//*[@id="d_chart_search"]/div/div/div[1]/div[1]/ul/li[1]/span/label'
age = d.find_element_by_xpath(age_xpath)
age.click()

f=open('melon_weekly.txt','w',-1,'UTF-8')

for i in range(9, 11): #연도 10번 선택

    try:
            year_xpath = '//*[@id="d_chart_search"]/div/div/div[2]/div[1]/ul/li[' + str(i) + ']/span/label' #년도 선택
            year = d.find_element_by_xpath(year_xpath)
            year.click()
            print(year.text)

    except:
            print("year_xpath not found")
            pass

    sleep(1)
    for i in range(1, 13): #월 12번 선택


        try:
            month_xpath = '//*[@id="d_chart_search"]/div/div/div[3]/div[1]/ul/li[' + str(i) + ']/span/label' #월선택
            month = d.find_element_by_xpath(month_xpath)
            month.click()
            print(month.text)

        except:
            print("month_xpath not found")
            pass

        sleep(1)
        for i in range(1, 6): #주간 5번 선택

            try:
                week_xpath = '//*[@id="d_chart_search"]/div/div/div[4]/div[1]/ul/li[' + str(i) + ']/span/label' #주선택
                week = d.find_element_by_xpath(week_xpath)
                week.click()
                print(week.text)

            except:
                print("week_xpath not found")
                break
```

selenium의 webdriver 기능을 활용하여 크롬브라우저에서 자동적으로 차트파인더의 항목을 클릭하고, beautifulsoup의 기능을 활용하여 웹페이지의 html정보를 파싱하였습니다. 첫 for 문에서 연도, 그 안 for문에서 월, 그 안 for문에서 주를 선택합니다. 주나 월이 존재하지 않을 시 반복문을 탈출하는 예외처리도 하였습니다. webdriver의 xpath기능을 활용해 반복적으로 원하는 항목을 선택하여 자동으로 클릭하는 기능을 합니다.

```
classCd = d.find_element_by_xpath('//label[@for = "gnr_1"]')
if '가요' not in classCd.text:
    classCd = d.find_element_by_xpath('//label[@for = "gnr_2"]')

classCd.click()
sleep(1)

d.find_element_by_xpath('//*[@id="d_srch_form"]/div[2]/button/span/span').click() #검색 버튼 누르기
sleep(3)

artists = d.find_elements_by_xpath('//*[@id="lst50"]/td[4]/div/div/div[2]/div[1]/a[1]')

changed_ranks = d.find_elements_by_xpath('//*[@id="lst50"]/td[2]/div/span[3]/span[1]') #어떻게 변동되었는지 (상승,하락,유지,NEW)

#변동된 순위, NEW(차트진입)인 경우 변동된 순위가 없어서 changed_ranks_num과 changed_ranks의 인덱스가 서로 mapping되지 않기 때문
changed_ranks_num = list()

a=0
for i,c in zip(range(0,50),changed_ranks):
    temp = d.find_elements_by_xpath('//*[@id="lst50"]/td[2]/div/span[3]/span[2]')
    if 'NEW' in c.text:
        changed_ranks_num.append(str(0))
        a=a+1 #NEW가 몇개있나
    else : changed_ranks_num.append(temp[i-a].text)

for i,artist,changed_rank, changed_rank_num in zip(range(1,51), artists, changed_ranks, changed_ranks_num):#1위부터 50위까지에

    t = "" #곡명 저장하는 변수

    try:
        t = d.find_element_by_xpath('//*[@id="lst50"][' + str(i) + ']/td[4]/div/div/div[1]/span/strong/a').text #곡명 추가
    except:
        t = d.find_element_by_xpath('//*[@id="lst50"][' + str(i) + ']/td[4]/div/div/div[1]/span/strong/span').text #재생불가능

    if '19금' in t:
        t = t.replace('19금', '')

    print("차트 연도:", year.text)
    print("달:", month.text)
    print("주:", week.text)
    print("순위:", i)
    print("제목:", t)
    print("아티스트:", artist.text)
    print(changed_rank_num + ' ' + changed_rank.text)
    print("----")

    data = "{0}//{1}//{2}//{3:3d}//{4}//{5}//{6}//{7}".format(re.sub('[^0-9]', '', year.text),month.text,week.text,i,t,artist.te
    f.write(data + '₩n')


d.find_element_by_xpath('//*[@id="frm"]/div[2]/span/a').click() #51-100위권 페이지 진입
sleep(3)
```

주간 차트 for문의 세부 코드입니다. 주간차트를 클릭한 후 차트의 장르를 선택한 후 검색을 누르면 화면상에 50위까지만 출력되고 51위부터 100위까지는 더보기 버튼을 클릭해야 볼 수 있습니다. 그렇기 때문에 저는 1위부터 50위까지를 먼저 출력, 51위-100위 보기 버튼을 클릭한 후 위의 과정을 반복하도록 작성하였습니다. 먼저 차트의 가수들을 xpath기능을 활용해 순위 순서대로 artist에 저장합니다. Find_elements_by_xpath함수를 사용했습니다. 다음으로는 changed_ranks에 마찬가지로 순위대로 상태를 저장합니다(순위 상승,순위 하락,동일,NEW 4가지). 다음으로는 변동된 순위가 어느정도인지 저장하는 changed_ranks_num 배열을 선언하였는데, 상태가 NEW인 경우에 변동된 순위의 값이 NULL이었기 때문에 changed_ranks와 chaned_ranks_num의 인덱스가 서로 매칭되지 않는 오류가 있었기 때문에 "NEW"인 경우 changed_ranks_num에 0을 저장해주는 오류 처리를 하였고, 그 밑 zip을 사용한 for문은 50위까지의 data를 출력하는 부분입니다. title에 대한 오류 - 멜론에서 재생할

수 없는 노래의 xpath가 일반 노래 제목의 xpath와 다른 문제 때문에 title에 대한 예외처리도 필요했습니다. 밑으로 제가 수집한 정보들을 jupyter notebook상에서 출력하고, text파일에 '//'로 구분하여 저장하는 것을 끝으로 for문이 종료됩니다. '//'로 구분을 한 이유는 data frame을 처리할 때 split(',')를 하면 노래 제목에 , 이 들어간 경우 (ex. 폴킴 - 모든날, 모든 순간) 노래 제목도 분리되어 버렸기 때문에 '//'로 하였습니다.



위 사진은 jupyter notebook상에서의 출력 화면이고,

위 사진은 저장된 txt파일을 열었을 때 화면입니다.

초기 데이터인 txt파일의 용량은 총 4.13MB이고, Data Frame처리 과정에서 순위 변동지수에 대한 새로운 열을 추가하여 저장한 **정제된 데이터의 csv파일 용량은 5MB입니다.**

	A	B	C	D	E	F	G	H	I
1	2019	1	12.31~01.	1	180◆툇	躡◆	0	◆넚◆쐞	0
2	2019	1	12.31~01.	2	◆꼺爾맸쓽	◆뿞◆뵫◆	0	NEW	98
3	2019	1	12.31~01.	3	◆떝◆슛◆	◆븯◆◆	3	◆땚쭹◆	3
4	2019	1	12.31~01.	4	SOLO	◆젇◆땲(	0	◆넚◆쐞	0
5	2019	1	12.31~01.	5	◆븑◆굞◆	MINO (◆넚	3	◆땚쭹◆	-3
6	2019	1	12.31~01.	6	MILLIONS	WINNER	3	◆땚쭹◆	-3
7	2019	1	12.31~01.	7	躡뚯뜥 12	泥◆◆븯	0	NEW	93
8	2019	1	12.31~01.	8	◆꼩璨◆	◆룧◆궐	3	◆땚쭹◆	-3
9	2019	1	12.31~01.	9	YES or YE	TWICE (◆	2	◆땚쭹◆	-2
10	2019	1	12.31~01.	10	◆궐 ◆깭	耳◆◆씄◆	1	◆땚쭹◆	1
11	2019	1	12.31~01.	11	紐⑤뱭 ◆	◆룧◆궐	1	◆땚쭹◆	1
12	2019	1	12.31~01.	12	泥ル늿◆뽙	◆붘◆씄朴	4	◆땚쭹◆	-4
13	2019	1	12.31~01.	13	Love Shot	EXO	3	◆땚쭹◆	-3
14	2019	1	12.31~01.	14	媛◆◆쓣	諛붿씄釉◆	5	◆땚쭹◆	-5
15	2019	1	12.31~01.	15	◆굞◆굞	◆븑◆씄◆	2	◆땚쭹◆	-2
16	2019	1	12.31~01.	16	想죺갚	◆뻴◆땚◆	2	◆땚쭹◆	-2
17	2019	1	12.31~01.	17	◆쓷◆븳	◆봄媛◆	1	◆땚쭹◆	1

2) 빅데이터 분석 : 회귀 분석

① Data Frame 생성

txt 파일 불러와서 Df 생성

```
In [2]: mRdd=spark.sparkContext#
            .textFile(os.path.join("data","melon_weekly2019-2010.txt"))
        print (mRdd.first())
```

```
2019//01월//12.31~01.06//  1//180도//벤//0//순위 동일
```

```
In [3]: mRdd1=mRdd.map(lambda x: x.split('//'))
```

```
In [7]: from pyspark.sql import Row
        mRdd=mRdd1.map(lambda x:Row(int(x[0]),int(x[1].lstrip().rstrip(u'₩uc6d4')),x[2].lstrip(),int(x[3]),x[4].lstrip(),x[5].lstrip(),int(x[6].lst
        mDf=spark.createDataFrame(mRdd)
```

```
In [8]: mDf=mDf.withColumn("Year",mDf['_1'].cast("long")).drop('_1')
        mDf=mDf.withColumn("Month",mDf['_2'].cast("integer")).drop('_2')
        mDf=mDf.withColumn("Week",mDf['_3'].cast("string")).drop('_3')
        mDf=mDf.withColumn("Rank",mDf['_4'].cast("integer")).drop('_4')
        mDf=mDf.withColumn("Title",mDf['_5'].cast("string")).drop('_5')
        mDf=mDf.withColumn("Artist",mDf['_6'].cast("string")).drop('_6')
        mDf=mDf.withColumn("Changed_rank",mDf['_7'].cast("integer")).drop('_7')
        mDf=mDf.withColumn("State",mDf['_8'].cast("string")).drop('_8')
```

```
In [9]: mDf.show()
```

```
+----+-----+-------------+----+--------------------+----------+------------+-----+
|Year|Month|         Week|Rank|               Title|    Artist|Changed_rank|State|
+----+-----+-------------+----+--------------------+----------+------------+-----+
|2019|    1|12.31~01.06|   1|                180도|        벤|      이순위 동일|
|2019|    1|12.31~01.06|   2|              넘쳐흘러|  엠씨더맥스|          이 NEW|
|2019|    1|12.31~01.06|   3|              신용재|      하은|      3단계 상승|
|2019|    1|12.31~01.06|   4|              SOLO| 제니 (JENNIE)|      이순위 동일|
|2019|    1|12.31~01.06|   5|              아낙네|  MINO (송민호)|      3단계 하락|
|2019|    1|12.31~01.06|   6|           MILLIONS|    WINNER|      3단계 하락|
|2019|    1|12.31~01.06|   7|            벌써 12시|      청하|          이 NEW|
|2019|    1|12.31~01.06|   8|              너를 만나|      폴킴|      3단계 하락|
|2019|    1|12.31~01.06|   9|       YES or YES|TWICE (트와이스)|      2단계 하락|
|2019|    1|12.31~01.06|  10|          내 생에 아름다운|    케이윌|      1단계 상승|
|2019|    1|12.31~01.06|  11|모든 날, 모든 순간 (Ever...|      폴킴|      1단계 상승|
|2019|    1|12.31~01.06|  12|                첫눈에| 헤이즈 (Heize)|      4단계 하락|
|2019|    1|12.31~01.06|  13|           Love Shot|     EXO|      3단계 하락|
|2019|    1|12.31~01.06|  14|          가을 타나 봐|  바이브|      5단계 하락|
|2019|    1|12.31~01.06|  15|                삐삐|  아이유|      2단계 하락|
|2019|    1|12.31~01.06|  16|                고백|  양다일|      2단계 하락|
|2019|    1|12.31~01.06|  17|              흔한 이별|      허각|      1단계 상승|
|2019|    1|12.31~01.06|  18|               Tempo|     EXO|      2단계 하락|
|2019|    1|12.31~01.06|  19|         아름답고도 아프구나|  비투비|      4단계 하락|
```

앞서 txt파일로 저장한 melon_weekly2019~2010.txt를 Rdd로 불러와서 // 기준으로 split 후 공백문자와 month 데이터의 '월'을 rstrip()으로 제거 후 형변환을 거쳐 Row객체에 저장해 준 후 Data Frame으로 변환했습니다. 이때 열의 순서가 제멋대로 생성되었기 때문에 withColumn을 이용해 열에 이름을 붙여주고, 형변환도 정확하게 다시 해줌으로써 데이터의 정확성을 높혔습니다. 여기까지가 제가 크롤링 한 데이터를 분석할 수 있는 형태의 데이터 프레임으로 변환시키는 과정입니다.

생성한 df 바탕으로 분석에 필요한 순위 변동추이 계산하여 행 만들기

In [10]:
```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType
def classifyState(s,c,r):
    q=0
    if s==u"단계 하락":
        q=c*(-1)
    elif s==u"단계 상승":
        q=c
    elif s=="NEW":
        q=100-r
    elif s==u"순위 동일":
        q=0
    return q

state_udf = udf(classifyState, StringType())
```

In [11]:
```
mDf2=mDf.withColumn("Rank_variation", state_udf(mDf.State,mDf.Changed_rank,mDf.Rank))
mDf2.show()
mDf2.write.format('com.databricks.spark.csv').save(os.path.join('data','MelonWeekly.csv'))
```

```
+----+-----+-----------+----+--------------------+----------+------------+-----+-------------+
|Year|Month|       Week|Rank|               Title|    Artist|Changed_rank|State|Rank_variation|
+----+-----+-----------+----+--------------------+----------+------------+-----+-------------+
|2019|    1|12.31~01.06|   1|                 180도|       벤|      0|순위 동일|            0|
|2019|    1|12.31~01.06|   2|              넘쳐흘러|  엠씨더맥스|           0|  NEW|           98|
|2019|    1|12.31~01.06|   3|               신용재|      하은|           3|단계 상승|            3|
|2019|    1|12.31~01.06|   4|              SOLO| 제니 (JENNIE)|      0|순위 동일|            0|
|2019|    1|12.31~01.06|   5|               아낙네|  MINO (송민호)|           3|단계 하락|           -3|
|2019|    1|12.31~01.06|   6|            MILLIONS|    WINNER|           3|단계 하락|           -3|
|2019|    1|12.31~01.06|   7|              벌써 12시|      청하|           0|  NEW|           93|
|2019|    1|12.31~01.06|   8|               너를 만나|     폴킴|           3|단계 하락|           -3|
|2019|    1|12.31~01.06|   9|          YES or YES|TWICE (트와이스)|           2|단계 하락|           -2|
|2019|    1|12.31~01.06|  10|           내 생에 아름다운|    케이윌|           1|단계 상승|            1|
|2019|    1|12.31~01.06|  11|모든 날, 모든 순간 (Ever...|     폴킴|           1|단계 상승|            1|
|2019|    1|12.31~01.06|  12|                첫눈에| 헤이즈 (Heize)|      4|단계 하락|           -4|
|2019|    1|12.31~01.06|  13|           Love Shot|     EXO|           3|단계 하락|           -3|
|2019|    1|12.31~01.06|  14|            가을 타나 봐|    바이브|           5|단계 하락|           -5|
|2019|    1|12.31~01.06|  15|                삐삐|    아이유|           2|단계 하락|           -2|
|2019|    1|12.31~01.06|  16|                고백|    양다일|           2|단계 하락|           -2|
```

제가 분석할 정보는 순위의 변동 추이이기 때문에, Udf 패키지를 활용하여 State에 따라 순위 변동계수를 설정해주는 함수를 작성했습니다. 단계 하락시에는 하락된 순위 만큼 음수로 취하고, 단계가 상승된 경우에는 상승 순위 그대로, 순위가 동일한 경우에는 0, NEW로 차트에 진입한 곡인 경우에 100위권 밖에서 상승한 것으로 취급하여 100-순위 만큼의 순위변동계수를 설정해주었습니다. 이 함수를 적용하여 Rank_Variation이라는 새로운 열을 생성하여 최종적으로 제가 사용할 데이터 프레임을 만들었고 이를 쉽게 볼 수 있게 csv파일로 저장하였습니다. 하지만 csv파일에서는 한글을 제대로 저장할 수 없다는 단점이 있었습니다.

② 그래프로 시각화

```
In [14]: mDf3=mDf2.groupBy('Month').agg({"Rank_variation":"sum"}).sort('Month')
         mDf3.show(3)
```

```
+-----+--------------------+
|Month|sum(Rank_variation)|
+-----+--------------------+
|    1|              6409.0|
|    2|              5128.0|
|    3|              7638.0|
+-----+--------------------+
only showing top 3 rows
```

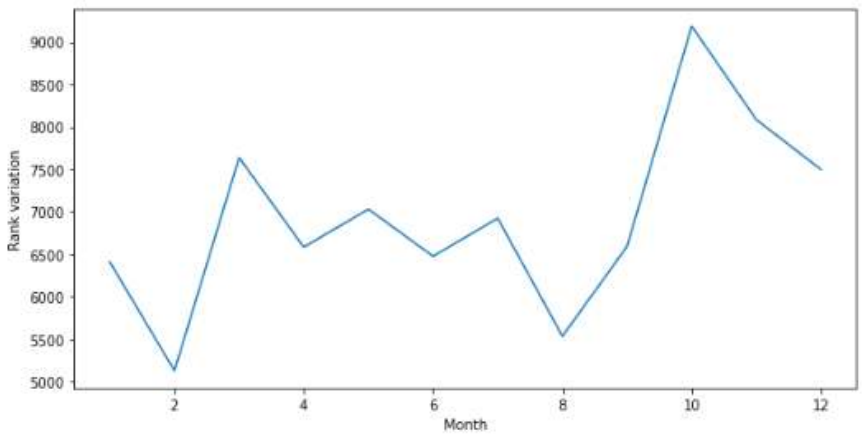
```
In [13]: import pandas as pd
         mDf4=mDf2.groupBy('Year').pivot('Month').agg({"Rank_variation":"sum"}).sort('Year')
         mDf4.show()
         pdf=mDf4.toPandas()
         pdf.head()
```

```
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+------+------+------+
|Year|    1|    2|    3|    4|    5|    6|    7|    8|    9|    10|    11|    12|
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+------+------+------+
|2010|830.0|473.0| 719.0| 861.0|1103.0| 599.0|1365.0|684.0|1112.0|1325.0|1095.0|1302.0|
|2011|918.0|800.0|1327.0| 736.0|1138.0|1414.0|1151.0|785.0| 857.0|1223.0|1212.0|1044.0|
|2012|707.0|704.0|1365.0|1065.0|1159.0|1067.0| 766.0|669.0| 998.0|1222.0|1113.0| 751.0|
|2013|991.0|938.0| 989.0| 892.0| 906.0| 743.0| 757.0|820.0| 754.0|1202.0| 857.0| 977.0|
|2014|932.0|792.0| 693.0| 796.0| 888.0| 645.0| 871.0|677.0| 552.0|1309.0| 979.0| 538.0|
|2015|819.0|427.0| 608.0| 742.0| 495.0| 680.0| 851.0|523.0| 589.0| 794.0| 686.0|1545.0|
|2016|475.0|371.0| 745.0| 588.0| 627.0| 754.0| 612.0|415.0| 820.0| 847.0| 559.0| 830.0|
|2017|290.0|287.0| 319.0| 238.0| 232.0| 339.0| 169.0|279.0| 384.0| 342.0| 568.0| 323.0|
|2018|146.0|192.0| 437.0| 322.0| 260.0| 137.0| 146.0|382.0| 178.0| 380.0| 663.0| 187.0|
|2019|301.0|144.0| 436.0| 344.0| 223.0|  98.0| 236.0|300.0| 348.0| 543.0| 354.0|  null|
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+------+------+------+
```

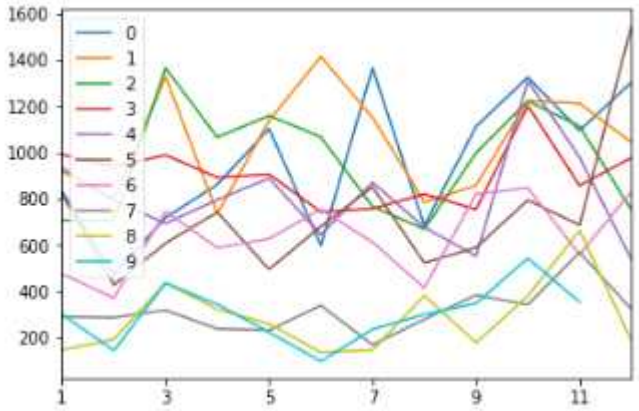
Out[13]:

	Year	1	2	3	4	5	6	7	8	9	10	11	12
0	2010	830.0	473.0	719.0	861.0	1103.0	599.0	1365.0	684.0	1112.0	1325.0	1095.0	1302.0
1	2011	918.0	800.0	1327.0	736.0	1138.0	1414.0	1151.0	785.0	857.0	1223.0	1212.0	1044.0
2	2012	707.0	704.0	1365.0	1065.0	1159.0	1067.0	766.0	669.0	998.0	1222.0	1113.0	751.0
3	2013	991.0	938.0	989.0	892.0	906.0	743.0	757.0	820.0	754.0	1202.0	857.0	977.0
4	2014	932.0	792.0	693.0	796.0	888.0	645.0	871.0	677.0	552.0	1309.0	979.0	538.0

위의 정제된 데이터 프레임에서 원하는 정보만을 추출해 두가지 방식으로 나타내었습니다. 위의 **mDf3는 월별 Rank variation의 합**, 밑의 **mDf4는 연도별 월을 pivot으로 하여 Rank variation의 합**을 나타낸 자료이고, pandas로도 한번 더 시각화 하였습니다. 저의 회귀 분석에서는 mDf4의 자료가 사용됩니다.

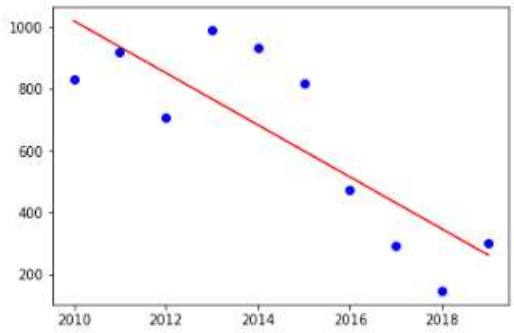


위 그래프는 mDf3의 그래프입니다. X축은 월, Y축은 Rank variation입니다. 주로 3월,10월쯔음 순위 변동추이가 상승하는 것을 이 그래프를 통해 확인할 수 있습니다. 또한 연말로 갈수록 순위 변동이 크다는 것 또한 확인할 수 있습니다.

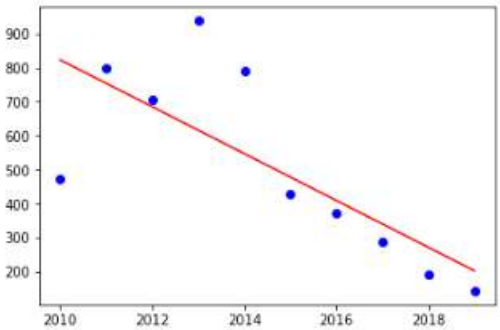


위 그래프는 mDf4의 그래프입니다. 0=2010,9=2019인데 2010년대 초반에는 순위 변동 추이의 변화가 다소 급격하고, 순위의 당락의 폭이 크다는 것 또한 알 수 있습니다. 후반대로 갈수록 그래프의 변화도 안정적이고, 순위의 변동이 크지 않습니다.

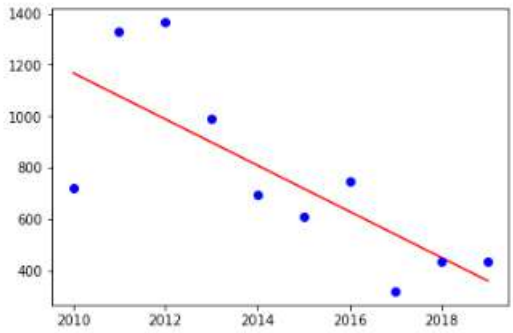
③ 회귀 분석



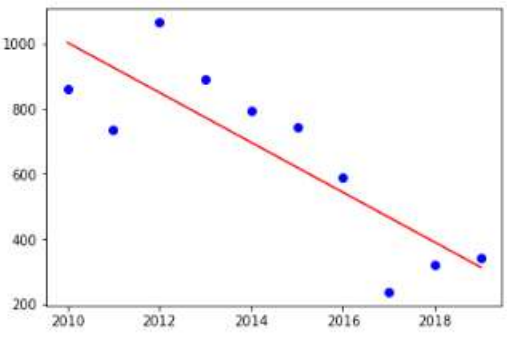
1월



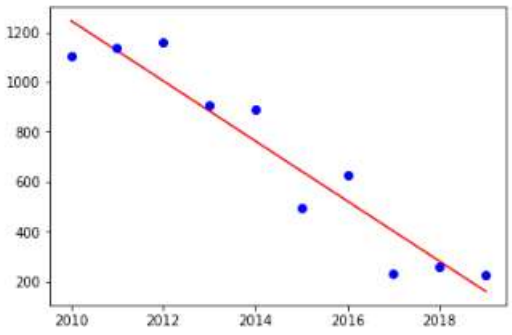
2월



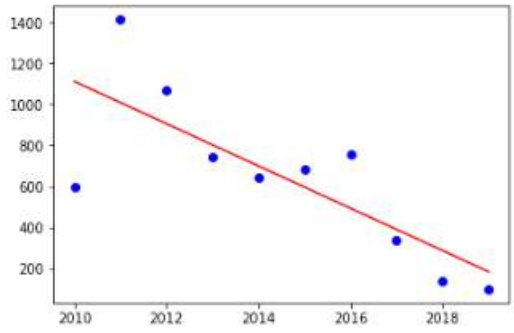
3월



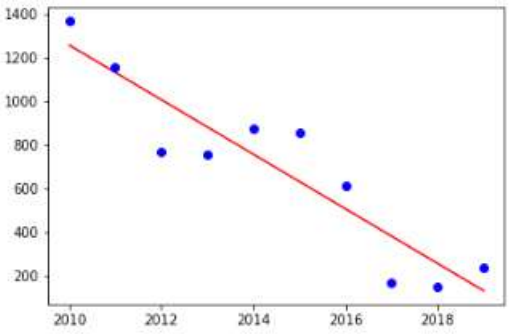
4월



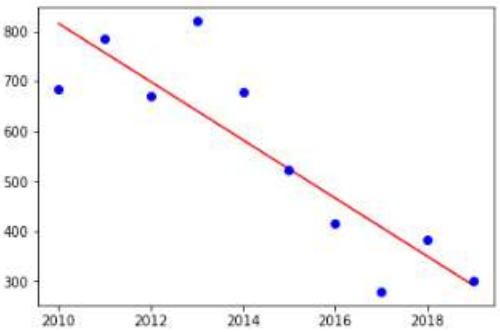
5월



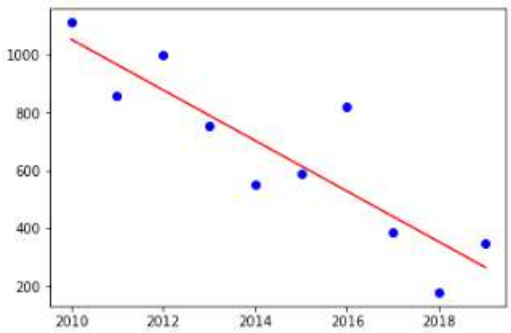
6월



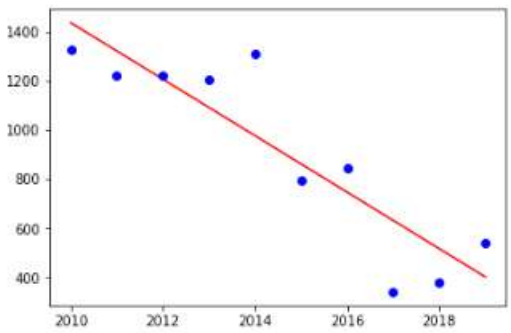
7월



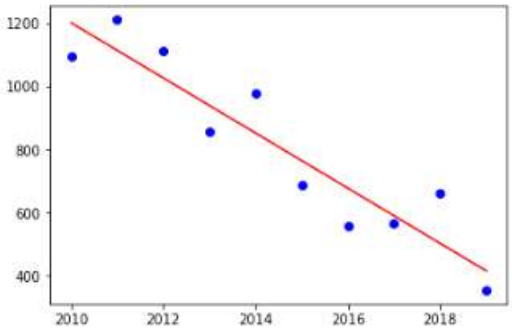
8월



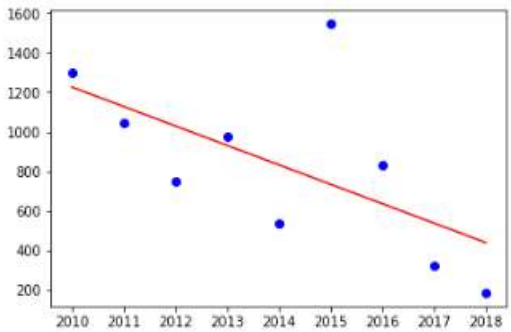
9월



10월



11월



12월

위 그래프들은 아래 코드를 바탕으로 회귀분석을 실시해 나타낸 결과입니다. X축은 연도, Y축은 rank variation이며 점들은 매년 해당하는 월의 rank_variation의 sum값을 나타내며, 회귀분석으로 찾아낸 yhat의 값은 빨간 실선으로 표시하였습니다.

Regression Analysis

In [16]:
```
#1월부터 12월까지, 매년 순위변동추이의 값의 변화를 scatter plot으로 확인. X축은 연도 Y축은 순위변동정도

X=mDf4.select('Year').collect()
Y=list()

print (X)
for i in range(1,13):
    Y.append([mDf4.select(str(i)).collect()])
    print(Y[i-1])
    plt.scatter(X,Y[i-1])
    plt.show()
```

Y배열에 매 월 rank_variation을 저장하는 이차원 배열을 만드는 코드

In [17]:
```
import statsmodels.api as sm
import sympy as sp
B0=list()
B1=list()
R2=list()
def regressionAnalysis(month) :

    Xt=np.transpose(X)[0]

    A=np.array([Xt,np.ones(len(Xt))])

    Yt=np.transpose(Y[month-1][0])[0]

    print month
    b0,b1=np.linalg.lstsq(A.T,Yt)[0]
    yhat=b0*Xt+b1
    B0.append(b0)
    B1.append(b1)
    plt.plot(Xt,yhat,'r-',Xt,Yt,'bo')
    plt.show()

    result = sm.OLS(Yt,A.T).fit()
    R2.append(result.rsquared)

for month in range(1,12): #11월까지 매년 순위변동추이
    regressionAnalysis(month)

#아직 차트가 존재하지 않는 2019년 12월
Xt=np.transpose(X)[0][:6]
A=np.array([Xt,np.ones(len(Xt))]).astype(np.float32)
Yt=np.transpose(Y[11][0])[0][:6].astype(np.float32)

b0,b1=np.linalg.lstsq(A.T,Yt)[0]
B0.append(b0)
B1.append(b1)
yhat=b0*Xt+b1
result = sm.OLS(Yt,A.T).fit()
R2.append(result.rsquared)

plt.plot(Xt,yhat,'r-',Xt,Yt,'bo')
plt.show()
```

회귀분석 함수의 내용입니다. 1월부터 11월까지의 for문에서 각각의 B0와 B1값을 리스트에 저장, Rsquared값을 계산하여 R2배열에 저장하였습니다. 2019년 12월의 데이터는 아직 없기 때문에 다른 월들과 데이터의 개수가 달라 반복문 밖에서 따로 작성하였습니다.

3) 결과

순위 예측

```
In [19]: def predictRankVariation(year,month):
             rv=0
             rv=B0[month-1]+year+B1[month-1]

             print "순위 변동추이: {0}, Rsquared값: {1}".format(rv,"%.2f"%R2[month-1])

In [20]: predictRankVariation(2019,12) #2019년 12월의 순위 변동추이는?
         predictRankVariation(2020,10) #2020년 10월의 순위 변동추이는?
         predictRankVariation(2020,1) #2020년 1월의 순위 변동추이는?

         순위 변동추이: 341.591651917, Rsquared값: 0.37
         순위 변동추이: 288.066666667, Rsquared값: 0.79
         순위 변동추이: 177.2, Rsquared값: 0.68

In [27]: sum=0
         for i in range(0,12):
             print "{0}월 회귀분석 Rsquared값 {1}".format(i+1,"%.2f"%R2[i])
             sum+=R2[i]
         print ("avarage R2:",sum/12)

         1월 회귀분석 Rsquared값 0.68
         2월 회귀분석 Rsquared값 0.56
         3월 회귀분석 Rsquared값 0.57
         4월 회귀분석 Rsquared값 0.71
         5월 회귀분석 Rsquared값 0.91
         6월 회귀분석 Rsquared값 0.61
         7월 회귀분석 Rsquared값 0.85
         8월 회귀분석 Rsquared값 0.78
         9월 회귀분석 Rsquared값 0.78
         10월 회귀분석 Rsquared값 0.79
         11월 회귀분석 Rsquared값 0.86
         12월 회귀분석 Rsquared값 0.37
         ('avarage R2:', 0.7047055837687015)
```

회귀 분석에서 얻은 B0,B1,R2값을 바탕으로 순위를 예측할 수 있는 predictRankVariation 함수를 만들었습니다. 년도와 월을 입력하면 순위 변동의 예측값과 그에대한 Rsquared값을 반환합니다. 예시로 실행을 해보았는데, 2019년 12월의 순위 변동추이 예상값은 341, R값은 0.37/ 2020년 10월의 예상값은 288, R값은 0.79/ 2020년 1월의 예상값은 177, R값은 0.68이었습니다. 마지막으로 모든 Rsquared값과 값의 평균을 출력했습니다. 가장 R값이 높았던 달은 5월 0.91의 R값을 보여줬고, 평균 Rsquared값은 0.70입니다.

문제점 : 2000년대 후반 이후에 들어서서야 스트리밍과 음원이 보급화 되었기 때문에 신뢰할 만한 데이터가 많이 쌓이지 않음 ,R값은 평균 0.7이지만 1월달의 경우 0.37의 저조한 R값을 보여줌, 또한 정확한 월별 순위 변동지수가 다른 요인을 밝혀내는 과정이 필요하며, 월별 신곡의 발매수가 순위 변동추이의 요인으로 작용할 수도 있음.

도출할 수 있는 결론 : 2010년대 후반으로 갈수록 월별 순위 변동지수가 낮아지고 있다. 이는 쟁쟁한 신곡이 많아 순위의 당락이 심했던 2010년도 초반에 비해 순위 변동이 안정되고 있다는 의미, 마찬가지로 한 노래가 오래 차트에 머물러 있는 추세라고도 할 수 있다. 그렇기 때문에 음반 제작사의 입장에서 롱런할 수 있는 가수의 음원 발매시기를 피하고, 예측 순위 변동지수가 클 때 신인 가수의 음반을 발매하거나, 순위 변동이 적은 시기에 음원 강자로 유명한 가수의 음원을 발매해 롱런을 꾀하는 등 위 자료를 통해 음원 발매 플랜을 계획하는 데 도움이 될 수 있음.