프로젝트 결과 보고서

프로젝트 주제	실시간 주가 데이터 수집 서버 구축과 ML을 통한 이상현상 감지
팀 명	소심이(5조)
프로젝트 기간	2021년 1월 25일 ~ 2021년 2월 3일
Special Thanks to 노태상 ♡	

조 원			
	이름	주 업무	보조 업무
1	박상균	데이터 서버 구축	데이터 수집&정제&가공 / 데이터 시각화 / 보고서 작성
2	강선미	데이터 분석 / 보고서 작성	데이터 구축 / 데이터 정제&가공 / 데이터 시각화 /보고서 작성
3	이아림	데이터 수집&정제&가공 / 데이터 시각화	데이터 구축 /보고서 작성
4	최정현		

1. 추진배경 및 필요성

작전주란? 증권브로커와 큰손, 대주주 등이 공모해 특정 기업의 주식을 매입, 주식값을 폭등시켜 이익을 챙기는 주가조작 행위를 말한다.

증권브로커들이 선정된 작전주를 사 모은 후, 일단 충분한 물량이 확보되면 자기들끼리 서로 사고팔면서 주식값을 올린다. 이때 주식값이 오르는 것을 보고 엉겁결에 작전주에 참여한 일반투자가들은 작전세력들이 가지고 있는 물량이 한꺼번에 쏟아지고 여기에 기관투자가까지 가세해 주가는 폭락, 큰 손해를 입는다.

작전 세력은 공정한 주식시장을 해칠 뿐만 아니라 주식 구매자들의 손해를 일으켜 사전에 발견할 필요가 있다.

2. 프로젝트 목표

: 실시간 주가를 가져오는 서버 구축. 작전 세력 투입 등 이상 현상 감지

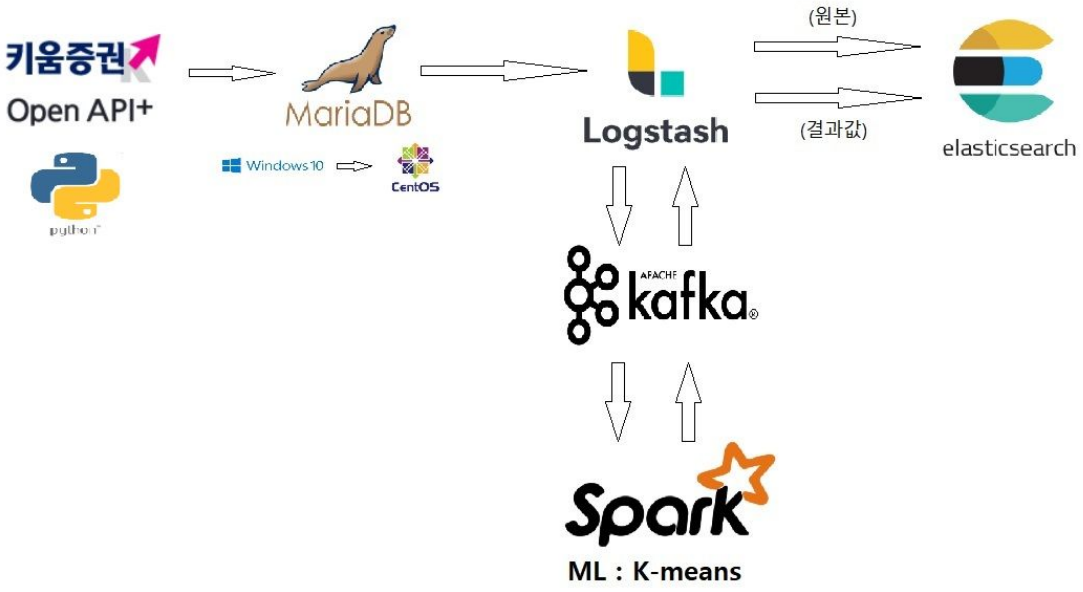
① 카더라 통신을 통해 일명 '작전주'라 불리는 종목 선정.

② 해당 종목의 1분당 주가 등락폭 / 1분당 거래량 폭을 가져오는 서버 구축.

③ K-means로 정상 범주의 주가 흐름과 비정상 범주의 주가 흐름 구분.

④ 실시간 데이터와 연동해 작전세력의 주식 매매 등 특이사항이 발생 감지

3. 프로젝트 내용



3 - 0 .주식 종목 선정

인터넷 커뮤니티, 카카오톡의 주식 리딩방 등 '카더라 통신'을 통해 최근 주가 조작 세력이 투입된 것 같다는 주식 정목 정보를 수집했다. 그 중 거래량이 많은 주식 종목 1개를 선정.

3 - 1 주가 데이터 파이썬 크롤링 - 키움증권API

키움증권 API는 리눅스 환경에서는 구동되지 않는다. 윈도우 OS에서 키움증권 API를 돌려서 데이터값을 받은 뒤 리눅스 mariaDB로 넘기는 방법을 택했다.

① 윈도우 파이썬에서 키움증권API와 연결, 해당 종목에 대한 1분당 주가 변동폭, 거래량 변동폭을 가져왔다.

```
def opt10080(self, rqname, trcode):
    data_cnt = self.get_repeat_cnt(trcode, rqname)
    self.create_table()
    p0, v0 = 0, 0
    for i in range(data_cnt):
        print('data', i)
        column = ["체결시간", "현재가", "거래량"]
        value = ['', '', '']
        for idx, col in enumerate(column):
            value[idx] = self.get_comm_data(trcode, rqname, i, col)
        rows = [value[0][:8], value[0][8:]]
        rows.append((int(p0)-int(value[1]))/int(value[1])*100)
        rows.append((int(v0)-int(value[2]))/int(value[2])*100)
        rows.append(code)

        p0, v0 = value[1], value[2]

        if self.flag != 0:
            print(rows)
            self.db_cur.execute(sql_insert, rows)
            self.db_conn.commit()
```

② pymysql.connect를 이용해 윈도우 파이썬 → CentOS mariadb로 전송.

```
# database
def create_table(self):
    self.db_conn = pymysql.connect(host='192.168.56.101', port=3306, user='root', password='[redacted]', db='mysql', ₩
                                   charset='utf8', database='stock')
    self.db_cur = self.db_conn.cursor()
    self.db_cur.execute(sql)
```

③ CentOS의 mariaDB에 데이터가 들어온 것을 확인

```
| 20210202 | 151900 |         0 |        129 | 038530 |
+----------+--------+-----------+------------+--------+
75165 rows in set (0.131 sec)

MariaDB [kiwoom3]> ;select * from trainingset limit 10;
ERROR: No query specified

+----------+--------+-----------+------------+--------+
| date     | time   | per_price | per_volume | code   |
+----------+--------+-----------+------------+--------+
| 20200203 | 090000 |        -4 |        -98 | 038530 |
| 20200203 | 090100 |         0 |         67 | 038530 |
| 20200203 | 090200 |         0 |       1061 | 038530 |
| 20200203 | 090300 |         3 |         62 | 038530 |
| 20200203 | 090400 |        -2 |        -83 | 038530 |
| 20200203 | 090500 |        -1 |        -98 | 038530 |
| 20200203 | 090600 |         1 |         24 | 038530 |
| 20200203 | 090700 |        -2 |       3448 | 038530 |
| 20200203 | 091000 |         1 |       -100 | 038530 |
| 20200203 | 091100 |         0 |        700 | 038530 |
+----------+--------+-----------+------------+--------+
10 rows in set (0.000 sec)
```

총 75,265개의 데이터가 들어왔고, 해당 종목의 약 1년간의 주가 데이터다.

3 - 2 Maira DB → logstash 데이터 전송

logstash에서 conf 파일을 작성, 실행해 데이터를 전송했다. output은 2가지 경로.
1 => elasticsearch로 index명은 'kiwoom_5+YYYYMMdd'로 전송
2 => kafka로 topic_id 'kiwoom'으로 전송

```
# mariaDB connect logstash

input {
  jdbc {
    jdbc_driver_library => "/lib/mysql-connector-java-8.0.18.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    jdbc_connection_string => "jdbc:mysql://localhost:3306/db_kiwoom?serverTimezone=Asia/Seoul"
    jdbc_user => "root"
    jdbc_password =>
    statement => "SELECT * FROM pastdata"
  }
}

output {
  stdout { codec =>  "rubydebug"}
  elasticsearch {
      index => "kiwoom_%{+YYYYMMdd}"
      hosts => ["localhost:9200"]
  }
  kafka {
    bootstrap_servers => "localhost:9092"
    topic_id => ["kiwoom"]
      codec => json
  }
}
```

3 - 3 kafka → spark 데이터 전송

```
scala> parsing.show()
+----+--------------------+------+---------+------+--------------------+-------------+
| key|               value| topic|partition|offset|           timestamp|timestampType|
+----+--------------------+------+---------+------+--------------------+-------------+
|null|{"low":-83800,"ti...|kiwoom|        0|     0|2021-02-02 16:35:...|            0|
|null|{"low":-83900,"ti...|kiwoom|        0|     1|2021-02-02 16:35:...|            0|
|null|{"low":-83900,"ti...|kiwoom|        0|     2|2021-02-02 16:35:...|            0|
|null|{"low":-83900,"ti...|kiwoom|        0|     3|2021-02-02 16:35:...|            0|
|null|{"low":-83900,"ti...|kiwoom|        0|     4|2021-02-02 16:35:...|            0|
|null|{"low":-83900,"ti...|kiwoom|        0|     5|2021-02-02 16:35:...|            0|
|null|{"low":-83800,"ti...|kiwoom|        0|     6|2021-02-02 16:35:...|            0|
|null|{"low":-83800,"ti...|kiwoom|        0|     7|2021-02-02 16:35:...|            0|
|null|{"low":-83800,"ti...|kiwoom|        0|     8|2021-02-02 16:35:...|            0|
|null|{"low":-83800,"ti...|kiwoom|        0|     9|2021-02-02 16:35:...|            0|
|null|{"low":-83700,"ti...|kiwoom|        0|    10|2021-02-02 16:35:...|            0|
|null|{"low":-83600,"ti...|kiwoom|        0|    11|2021-02-02 16:35:...|            0|
|null|{"low":-83600,"ti...|kiwoom|        0|    12|2021-02-02 16:35:...|            0|
|null|{"low":-83600,"ti...|kiwoom|        0|    13|2021-02-02 16:35:...|            0|
|null|{"low":-83400,"ti...|kiwoom|        0|    14|2021-02-02 16:35:...|            0|
|null|{"low":-83500,"ti...|kiwoom|        0|    15|2021-02-02 16:35:...|            0|
|null|{"low":-83500,"ti...|kiwoom|        0|    16|2021-02-02 16:35:...|            0|
|null|{"low":-83500,"ti...|kiwoom|        0|    17|2021-02-02 16:35:...|            0|
|null|{"low":-83500,"ti...|kiwoom|        0|    18|2021-02-02 16:35:...|            0|
|null|{"low":-83600,"ti...|kiwoom|        0|    19|2021-02-02 16:35:...|            0|
+----+--------------------+------+---------+------+--------------------+-------------+
only showing top 20 rows
```

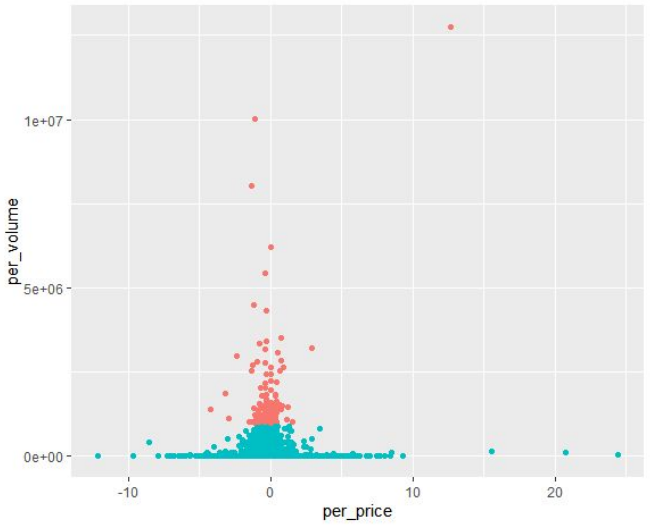
3 - 4 spark에서 K-means ML 분석

지난 1년 간의 1분당 주가 등락폭, 1분당 거래량 폭 데이터를 K-means로 분석, 중간값을 얻어냈다. 정상과 비정상 2가지 범주로 나누고 해당 범주의 center값을 얻어냈다.

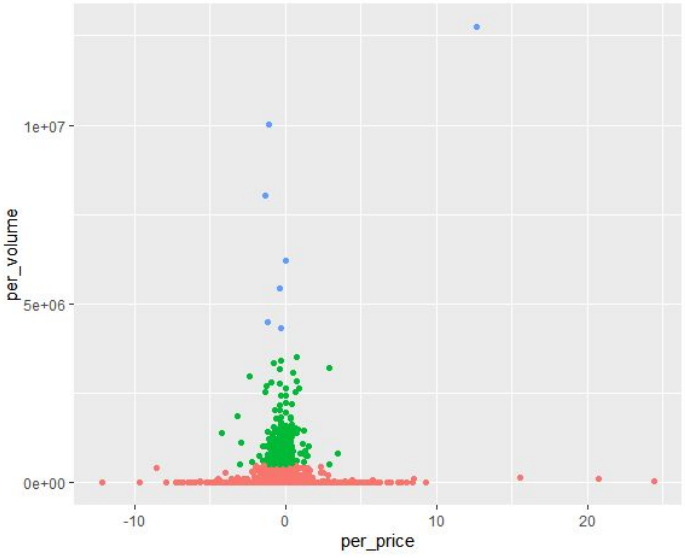
```
scala> featureDf.show(10)
+---------+----------+--------------+
|per_price|per_volume|      features|
+---------+----------+--------------+
|      0.0|      67.0|    [0.0,67.0]|
|      0.0|    1061.0|  [0.0,1061.0]|
|      3.0|      62.0|    [3.0,62.0]|
|     -2.0|     -83.0|  [-2.0,-83.0]|
|     -1.0|     -98.0|  [-1.0,-98.0]|
|      1.0|      24.0|    [1.0,24.0]|
|     -2.0|    3448.0| [-2.0,3448.0]|
|      1.0|    -100.0|  [1.0,-100.0]|
|      0.0|     700.0|   [0.0,700.0]|
|     -1.0|   14900.0|[-1.0,14900.0]|
+---------+----------+--------------+
only showing top 10 rows
```

```
scala> val kmeans = new KMeans().setK(2).setFeaturesCol("features").setPredictionCol("prediction")
kmeans: org.apache.spark.ml.clustering.KMeans = kmeans_cfbd5e672b0a

scala> val kmeansModel = kmeans.fit(featureDf)
kmeansModel: org.apache.spark.ml.clustering.KMeansModel = kmeans_cfbd5e672b0a

scala> kmeansModel.clusterCenters.foreach(println)
[-7.323669119310243,8818.794204132682]
[-30.857142857142854,7324071.428571428]
```

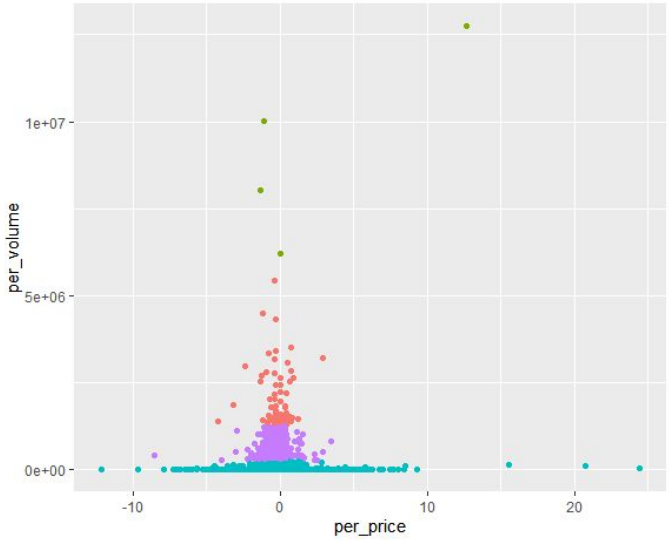
3 - 5 시각화




```
> dkmeans2$centers
     per_price  per_volume
1 -0.140843433 1847966.429
2  0.001945415    6420.482
```




```
> dkmeans3$centers
     per_price  per_volume
1  0.002402225    4826.091
2 -0.193725573  975386.261
3  1.203238036 7324071.429
```




```
> dkmeans4$centers
     per_price  per_volume
1 -0.180609933 2048969.403
2  2.564772314 9257825.000
3  0.003417292    3459.687
4 -0.232623725  526635.988
```