1)데이터 수집(크롤링) 모듈

모듈명	지니 차트 크롤링 모듈
기능	지니 차트의 1~100위까지의 곡이름,아티스트명,랭킹,날짜를 가져와 데이터베이스에 저장하는 모듈.

소스	#Genie_main.py
	from calendar import Calendar
	from Genie_to_Db import GenieTodB
	
	c = Calendar()
	year = 2018
	for i, month in enumerate(c.yeardayscalendar(year, width=1)):
	  for week_7 in month:
	  for week in week_7:
	  for date in week:
	  if date == 0:
	  continue
	  else:
	  #print(i+1, date)
	  if (i+1)//10 < 1:
	  si = '0'+str(i+1)
	  else:
	  si = str(i+1)
	  if date//10 < 1:
	  sdate = '0'+str(date)
	  else:
	  sdate = str(date)
	---------- 	
	#Genie_Chart_Crawling.py
	from bs4 import BeautifulSoup as bs
	import requests
	
	# getdata(160101, 1)
	def getData(ymd,pagenum):
	  url = 'http://www.genie.co.kr/chart/top200?ditc=D&rtm=N&ymd=20'+ymd+'&pg='+pagenum
	  with requests.get(url) as r:
	  r.encoding = 'UTF-8'
	  html = r.text

```
        soup = bs(html, 'html.parser')
        listbody = soup.find('tbody')
        songranks = listbody.find_all('td',{'class':'number'})
        titles = listbody.find_all('a', {'class':'title ellipsis'})
        artists = listbody.find_all('a', {'class':'artist ellipsis'})

        songrank = map(lambda x: ''.join(x.text.split()[0]), songranks)
        #rankwave = map(lambda x: ''.join(x.text.split()[-1]), songranks)
        titles = map(lambda x: ''.join(x.text.split()), titles)
        artists = map(lambda x: ''.join(x.text.split()), artists)
        for song in zip(songrank, titles, artists):
            yield song
    return

------------------------------------------------------------
#Genie_to_Db.py
import pymysql
from Genie_Chart_Crawling import getData

#GenieTodB()
def GenieTodB(ymd):
    # MySQL Connection 연결
    conn = pymysql.connect(host='zuzak.cvqcrkck1aqg.us-east-1.rds.amazonaws.com', user='getChan', password='cksdl951!!',db='zuzak', charset='euckr')
    # Connection 으로부터 Cursor 생성
    curs = conn.cursor()
    # SQL문 실행
    for pn in range(1, 3):
        for i in getData(ymd, str(pn)):
            #print(i)
            #print(ymd+hour, i[0], i[1], i[2], i[3])
            sql = "INSERT INTO zuzak.genie VALUES (%s, %s, %s, %s)"
            val = (ymd, i[0], i[1], i[2])
            curs.execute(sql, val)
            # 데이타 Fetch
```

	curs.fetchone()
	pass
	pass
	
	#커밋
	conn.commit()
	print(ymd+'complete')
	curs.close()
	# Connection 닫기
	conn.close()
	return
	
	# Connection 닫기

입력	시작연도, 종료연도.
출력	없음(곡이름,아티스트명,랭킹,날짜를 데이터베이스에 저장).

모듈명	벅스 차트 크롤링 모듈
기능	벅스 차트의 1~100위까지의 곡이름, 아티스트명, 랭킹,날짜를 가져와 데이터베이스에 저장하는 모듈.

소스	package DBP;
	
	import java.util.Scanner;
	import java.io.IOException;
	import java.sql.*;
	import org.jsoup.Jsoup;
	import org.jsoup.nodes.Document;
	import org.jsoup.nodes.Element;
	import org.jsoup.select.Elements;
	import java.util.Calendar;
	import java.util.Date;
	import java.util.GregorianCalendar;
	import java.util.Locale;
	import java.text.ParseException;
	import java.text.SimpleDateFormat;
	
	public class asdf {
	
	public static String doDateAdd(String dates) throws ParseException { // 날짜 계산

```
        Calendar cal = new GregorianCalendar(Locale.KOREA); // cal 객체생성
        SimpleDateFormat fm = new SimpleDateFormat("yyyyMMdd");// Dateform 생성

        Date date = fm.parse(dates);// 입력받은 문자열 date를 Date형식으료 변환
        cal.setTime(date);
        cal.add(Calendar.DAY_OF_YEAR, 1); // 하루를 더한다.
        String strDate = fm.format(cal.getTime());
        return strDate;
    }

    public static void main(String[] args) throws ParseException {
        // Jsoup를 이용해서 벅스 노래차트 크롤링

        Scanner sc = new Scanner(System.in);

        System.out.print("시작 YYMMDD 입력 : ");
        String chardate = sc.next();
        System.out.println("시작 HH 입력 : ");
        int hour = sc.nextInt();
        System.out.println("종룔 YYMMDD 입력 : ");
        String from_chardate = sc.next();

        String ranking[] = new String[101];
        int i = 1;
        int k = 1;
        String insert_ranking[] = new String[101];
        String insert_Y;
        String insert_title[] = new String[101];
        String insert_artist[] = new String[101];

        /*
         * System.out.print("제목 검색 : "); String Song_name=sc.next();
         */

        do {
            insert_Y = "" + chardate;
```

```
            insert_Y = insert_Y.substring(2,8);
            String url =
"https://music.bugs.co.kr/chart/track/realtime/total?chartdate=" +
chardate + "&charthour="
                    + hour;
            Document doc = null;

            try {
                doc = Jsoup.connect(url).get();
            } catch (IOException e) {
                e.printStackTrace();
            }

            // 주요 뉴스로 나오는 태그를 찾아서 가져오도록 한다.
            Elements element = doc.select("div.innerContainer"); //
innerContainer 자료 가져오기

            for (Element el : element.select("div.ranking")) { // 1~100위
노래 차트 저장
                ranking[i] = el.text();
                i++;
            }
            i = 1;

            for (int j = 1; j <= 100; j++) {
                if (j < 10) {
                    insert_ranking[j] = ranking[j].substring(0, 1);
                } else if (j >= 10 && j <= 99) {
                    insert_ranking[j] = ranking[j].substring(0, 2);
                } else {
                    insert_ranking[j] = ranking[j].substring(0, 3);
                }
            }

            for (Element el : element.select("p.title")) { // 1~100위 노래
제목 저장
                insert_title[k] = el.text();
                k++;
            }
            k = 1;
```

```
        for (Element el : element.select("p.artist")) { // 1~100위 노래가수 저장
            insert_artist[k] = el.text();
            k++;
        }
        k = 1;


/*              System.out.println(); System.out.println("1위부터 100위 노래\n"); // 1위부터 100위 노래출력
          System.out.println("============================================================");
            for(int z=1;z<=100;z++) { System.out.print(insert_ranking[z]+" ");
            System.out.print(insert_title[z]+" "); System.out.print(insert_artist[z]);
            System.out.println(); }*/


        // DB연결
        Connection conn = null;
        PreparedStatement pstmt = null;
        Statement stmt = null;
        ResultSet rs = null;

        try {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("Dvrver is Loadded");
            String Durl = "jdbc:mysql://zuzak.cvqcrkck1aqg.us-east-1.rds.amazonaws.com?useUnicode=true&characterEncoding=euc_kr";
            String user = "getChan";
            String pass = "cksdl951!!";
            conn = DriverManager.getConnection(Durl, user, pass);
            stmt = conn.createStatement();

            String sql = "insert into zuzak.new_table values(?,?,?,?)"; // mysql에 저장하기 위한 쿼리문
```

	pstmt = conn.prepareStatement(sql);
	
	for (int j = 1; j <= 100; j++) {
	pstmt.setString(1, insert_Y);
	pstmt.setString(2, insert_ranking[j]);
	pstmt.setString(3, insert_title[j]);
	pstmt.setString(4, insert_artist[j]);
	
	pstmt.executeUpdate();
	}
	System.out.println("레코드 추가 완료");
	
	} catch (Exception e) {
	System.out.println(e);
	} finally {
	try {
	if (rs != null)
	rs.close();
	if (stmt != null)
	stmt.close();
	if (conn != null)
	conn.close();
	} catch (SQLException e) {
	}
	}
	chardate = doDateAdd(chardate);
	} while (!(chardate.equals(from_chardate)));
	sc.close();
	
	}
	}

입력	시작연도, 시작시간. 종료연도.
출력	없음(곡이름, 아티스트명, 랭킹, 날짜를 데이터베이스에 저장).

모듈명	엠넷 차트 크롤링 모듈
기능	엠넷 차트의 1~100위까지의 곡이름,아티스트명,랭킹,날짜를 가져와 데이터베이스에 저장하는 모듈.

소스	import java.io.IOException;
	import org.jsoup.Jsoup;
	import org.jsoup.nodes.Document;
	import org.jsoup.nodes.Element;
	import org.jsoup.select.Elements;
	import java.util.Scanner;
	import java.util.Calendar;
	import java.util.Date;
	import java.util.GregorianCalendar;
	import java.util.Locale;
	import java.text.ParseException;
	import java.text.SimpleDateFormat;
	import java.sql.*;
	
	public class date {
	
	  public static String doDateAdd(String dates) throws ParseException { // 날짜 계산
	  Calendar cal = new GregorianCalendar(Locale.KOREA); // cal 객체생성
	  SimpleDateFormat fm = new SimpleDateFormat("yyMMdd");// Dateform 생성
	
	  Date date = fm.parse(dates);// 입력받은 문자열 date를 Date형식으료 변환
	  cal.setTime(date);
	  cal.add(Calendar.DAY_OF_YEAR, 1); // 하루를 더한다.
	  String strDate = fm.format(cal.getTime());
	  return strDate;
	  }
	
	  public static void main(String[] args) throws ParseException {
	  // 준비
	  Scanner sc = new Scanner(System.in);
	  System.out.println("음원 탐색 기간 설정 (언제부터? ex) yyMMDD)");
	  String to_Date = sc.next();

```
        System.out.println("음원 탐색 기간 설정 (언제까지? ex) yyMMDD)");
        String from_Date = sc.next();
        String Artist = "";
        String Title = "";



        String insert_ranking[] = new String[101];
        String insert_title[] = new String[101];
        String insert_artist[] = new String[101];

        // DB연동 준비
        Connection conn = null;
        PreparedStatement pstmt = null;

        // 차트 크롤링 및 데이터 베이스 저장
        do {

            String url = "http://www.mnet.com/chart/TOP100/20" + to_Date;
            String url2 = "http://www.mnet.com/chart/TOP100/20" + to_Date + "?pNum=2";

            Document doc = null;
            Document doc2 = null;

            try { // Jsoup
                doc = Jsoup.connect(url).get(); // 엠넷 1번페이지 (1~50)
                doc2 = Jsoup.connect(url2).get(); // 엠넷 2번페이지 (51위~100위)
            } catch (IOException e) {
                e.printStackTrace();
            }
            // 주요 차트를 나오는 태그를 찾아서 가져오도록 한다.
            Elements element = doc.select("div.MMLTable");
            Elements element2 = doc2.select("div.MMLTable");
            int i = 1;
            int cnt = 1; // 음원 차트 순위 카운팅

            // 크롤링 구간!
```

```
            for (Element el : element.select("div.MMLITitle_Box.info")) {
                Title = el.select("a.MMLI_Song").text();
                Artist = el.select("a.MMLIInfo_Artist").text();

                // 크롤링 성공하면 다음 if문 실행
                //System.out.println("날짜:" + to_Date + " 제목:" + Title
+ " 가수 :" + Artist + " 순위:" + cnt + "위");
                // 디비저장

                insert_ranking[i] = Integer.toString(cnt);
                insert_title[i] = Title;
                insert_artist[i] = Artist;
                i++;
                cnt++;
            }
            // 1번페이지에 곡이없으면 다음 크롤링 실행

            for (Element el : element2.select("div.MMLITitle_Box.info")) {
                Title = el.select("a.MMLI_Song").text();
                Artist = el.select("a.MMLIInfo_Artist").text();

                //System.out.println("날짜:" + to_Date + " 제목:" + Title
+ " 가수 :" + Artist + " 순위:" + cnt + "위");

                insert_ranking[i] = Integer.toString(cnt);
                insert_title[i] = Title;
                insert_artist[i] = Artist;

                i++;
                cnt++;
            }
            // 만약 두 페이지에(모든 순위권에) 곡이 없으면 디비 저장안하
고 차트에 없다는 로그 생성
                try {
                    String                    db_url                    =
"jdbc:mysql://zuzak.cvqcrkck1aqg.us-east-1.rds.amazonaws.com?us
eUnicode=true&characterEncoding=euc_kr";
                    String id = "getChan";
                    String pw = "cksdl951!!";
```

```
                    Class.forName("com.mysql.jdbc.Driver");
                    conn = DriverManager.getConnection(db_url, id,
pw);

                    String sql = "insert into zuzak.mnet values(?,?,?,?)";
                    pstmt = conn.prepareStatement(sql);

                    for (int j = 1; j <= 100; j++) {
                        pstmt.setString(1, to_Date);
                        pstmt.setString(2, insert_ranking[j]);
                        pstmt.setString(3, insert_title[j]);
                        pstmt.setString(4, insert_artist[j]);

                        pstmt.executeUpdate();
                     }

                    System.out.println("mnet테이블에 DB "+to_Date+"일
차트 저장."); // 성공시 메시지 출력

                } catch (Exception e) { // 예외가 발생하면 예외 상황을
처리한다.
                    e.printStackTrace();
                    System.out.println(to_Date+"mnet 테이블에 새로운 레
코드 추가에 실패했습니다.");
                } finally { // 쿼리가 성공 또는 실패에 상관없이 사용한 자
원을 해제 한다. (순서중요)
                    if (pstmt != null)
                        try {
                            pstmt.close();
                        } catch (SQLException sqle) {




                        } // PreparedStatement 객체 해제
                    if (conn != null)
                        try {
                            conn.close();
                        } catch (SQLException sqle) {
                        } // Connection 해제
```

	}
	
	to_Date = doDateAdd(to_Date); // 날짜 하루를 증가시킴
	
	} while (!(to_Date.equals(from_Date)));
	sc.close();
	}
	}

입력	시작연도, 종료연도.
출력	없음(곡이름,아티스트명,랭킹,날짜를 데이터베이스에 저장).

2)데이터베이스 모듈

모듈명	데이터베이스 질의 모듈
기능	쿼리를 입력하면 db에 있는 데이터를 pandas DataFrame형태로 반환하는 모듈.

소스	import pymysql
	import pandas as pd
	from matplotlib import pyplot as plt
	
	# 쿼리를 입력하면 db에 있는 데이터를 pandas DataFrame형태로 반환
	def dbQuery(query):
	
	conn = pymysql.connect(host='zuzak.cvqcrkck1aqg.us-east-1.rds.amazonaws.com', user='getChan', password='cksdl951!!',db='zuzak', charset='euckr')
	q = query
	try:
	with conn.cursor() as cursor:
	df = pd.read_sql(q, con=conn)
	cursor.fetchall()
	finally:
	conn.close()
	return df

입력	질의문.
출력	pandas DataFrame(데이터).

모듈명	데이터베이스 전처리 모듈
기능	pandas DataFrame를 사용자가 좀더 보기 쉽게 전처리(가공)과정을 거치는 모듈.

소스	import pymysql
	import pandas as pd
	from matplotlib import pyplot as plt
	
	# DataFrame 전처리 함수.
	def dfFilter(dataFrame):
	  df = dataFrame.set_index("YYMMDD")
	  df.ranking =pd.to_numeric(df.ranking)
	
	  return df

입력	pandas DataFrame(데이터).
출력	가공된 pandas DataFrame(데이터).

모듈명	데이터베이스 시각화 모듈
기능	DataFrame을 사용자가 한눈에 볼 수 있는 그래프 형태로 변환시켜주는 모듈.

소스	import pymysql
	import pandas as pd
	from matplotlib import pyplot as plt
	
	# dataFrame 시각화
	def dfPlot(dataFrame):
	  dataFrame.plot()
	  plt.gca().invert_yaxis()
	  plt.ylabel('ranking')
	  plt.show()

입력	pandas DataFrame(데이터).
출력	데이터를 시각화한 그래프.

3) 판별 속성 모듈

모듈명	다른 노래 차트 진입일 모듈
기능	해당 음원 이외에 같은 가수의 다른 음원의 차트 진입을 구하여 리턴하는 모듈.

소스	from dbQuery import dbQuery, dfFilter, dfPlot
	# 다른 노래 차트진입일
	def chartInDays(title, artist, dbname):
	# 쿼리문 입력
	query = " SELECT * FROM zuzak."+dbname+" where title != \""+title+"\" and artist = \""+artist+"\" and YYMMDD<= (select YYMMDD from zuzak."+dbname+" where title = \""+title+"\" order by YYMMDD limit 1); "
	
	Chartdf = dbQuery(query)
	Chartdf = dfFilter(Chartdf)
	
	# 쿼리결과 없으면
	if Chartdf.empty:
	return 0
	
	# 쿼리 결과
	else :
	return len(Chartdf)

입력	title, artist, 데이터가 있는 dbname(DB이름)
출력	차트 진입일

모듈명	랭킹 상승 평균치 모듈
기능	해당 음원의 랭킹 상승 평균치를 구하여 리턴해주는 모듈.

소스	from dbQuery import dbQuery, dfFilter, dfPlot
	# 랭킹 상승 평균치
	def rankIncreaseMean(title, dbname):
	# 쿼리문 입력
	query = """ SELECT * FROM zuzak."""+dbname+""" where title = \"""+title+"""\" and YYMMDD<=
	(select YYMMDD from zuzak."""+dbname+""" where title = \"""+title+"""\" and cast(ranking as unsigned) <=5 order by YYMMDD limit 1); """
	
	Chartdf = dbQuery(query)
	Chartdf = dfFilter(Chartdf)
	
	# 쿼리결과 없으면
	if Chartdf.empty:
	return 0
	
	# 쿼리 결과
	elif len(Chartdf) == 1 :
	sum = 100-Chartdf['ranking'].iloc[0]
	return sum
	
	else:
	sum = 0
	a = len(Chartdf)
	
	for i in range(1, a):
	num = (Chartdf['ranking'].iloc[i-1]) - (Chartdf['ranking'].iloc[i])
	sum = sum + num
	
	return sum/(a-1)

입력	title, dbname
출력	랭킹 상승 평균치

모듈명	5위 진입 전까지의 일수 모듈
기능	해당 음원의 5위 진입 전까지의 일수를 구하여 리턴하는 모듈.

소스	from dbQuery import dbQuery, dfFilter, dfPlot
	def rankInDays(title, dbname):
	  # 쿼리문 입력
	  query = """ SELECT * FROM zuzak."""+dbname+""" where title = \"""+title+"""\" and YYMMDD<=(SELECT YYMMDD FROM zuzak."""+dbname+""" where title = \"""+title+"""\" and cast(ranking as unsigned) <=5 order by YYMMDD limit 1); """
	  Chartdf = dbQuery(query)
	  Chartdf = dfFilter(Chartdf)
	
	  # 쿼리결과 없으면
	  if Chartdf.empty:
	  return 0
	
	  # 쿼리 결과
	  else :
	  return len(Chartdf)

입력	title, dbname
출력	5위 진입 전까지의 일수

4) 딥러닝 모듈

모듈명	신경망 학습 모듈
기능	음원에서 빼낸 속성들로 하여금 학습을 시키는 신경망 학습 모듈.

소스	from getAttribute import dbQuery, chartInDays, rankIncreaseMean, rankInDays
	import tensorflow as tf
	import numpy as np
	
	geniedf = dbQuery("SELECT * FROM zuzak.genie where YYMMDD = 180715 and ranking <= 15")
	bugsdf = dbQuery("SELECT * FROM zuzak.bugs where YYMMDD = 180710 and ranking <= 15")
	mnetdf = dbQuery("SELECT * FROM zuzak.mnet where YYMMDD = 180701 and ranking <= 15")
	
	# 데이터프레임 합치기
	df = geniedf.append(bugsdf)
	df = df.append(mnetdf)
	df.reset_index(inplace=True, drop=True)
	
	# 조작음원 = 1
	# 여기선 지나오다, way back home 조작으로 가정
	tmp = df[(df['title']=='WayBackHome') | (df['title']=='지나오다')]
	labels = np.zeros([45, 1])
	labels[tmp.index] = 1
	
	# 이전에 차트에 든 일수
	chartindays = np.fromiter(map(chartInDays, df['title'], df['artist'], ['genie']*len(df)), np.int32)
	
	# 랭킹 상승 평균값
	rankimean = np.fromiter(map(rankIncreaseMean, df['title'], ['genie']*len(df)), np.int32)
	rankimean = np.array((rankimean+1) / (chartindays+1))
	
	# top5 진입 일수
	rankindays = np.fromiter(map(rankInDays, df['title'], ['genie']*len(df)), np.int32)
	rankindays = np.array((rankimean + 1) / (chartindays+1))

```
# 데이터 정규하
chartindays = np.array((chartindays+0.01) / (chartindays+1))
chartindays

train_data = np.array([chartindays, rankimean, rankindays]).T
train_data

tf.reset_default_graph()
# [chartindays, rankimean, rankindays]
x_data = train_data
y_data = labels

#########
# 신경망 모델 구성
######
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

# W = tf.Variable(tf.random_uniform([3, 1], -1., 1.))
W1 = tf.Variable(tf.random_normal([3, 50]))
b1 = tf.Variable(tf.random_normal([50]))
L1 = tf.nn.sigmoid(tf.add(tf.matmul(X, W1), b1))

W2 = tf.Variable(tf.random_normal([50, 1]))
b2 = tf.Variable(tf.random_normal([1]))
L2 = tf.nn.sigmoid(tf.add(tf.matmul(L1, W2), b2))

model = L2

cost = tf.reduce_mean(tf.square(model-Y))
optimizer = tf.train.AdamOptimizer(learning_rate=0.02)
train_op = optimizer.minimize(cost)

# 신경망 모델 학습

sess = tf.Session()
saver = tf.train.Saver(tf.global_variables())
```

	ckpt = tf.train.get_checkpoint_state('./model')
	if ckpt and tf.train.checkpoint_exists(ckpt.model_checkpoint_path):
	  saver.restore(sess, ckpt.model_checkpoint_path)
	else:
	  sess.run(tf.global_variables_initializer())
	  for step in range(1000):
	  sess.run(train_op, feed_dict={X: x_data, Y: y_data})
	  print(step + 1, sess.run(cost, feed_dict={X: x_data, Y: y_data}))
	
	prediction = model
	target = Y
	print('예측값:', sess.run(prediction, feed_dict={X: x_data}))
	print('실제값:', sess.run(target, feed_dict={Y: y_data}))

입력	각 음원 차트의 판별 속성.
출력	학습된 신경망 모듈.

모듈명	신경망 테스트 모듈
기능	학습을 통해 만들어진 신경망을 테스트 해보는 모듈.

소스	from getAttribute import dbQuery, chartInDays, rankIncreaseMean, rankInDays
	
	import tensorflow as tf
	import numpy as np
	
	df = dbQuery("SELECT * FROM zuzak.bugs where ranking <=15 and YYMMDD = 180105")
	
	# 이전에 차트에 든 일수
	chartindays = np.fromiter(map(chartInDays, df['title'], df['artist'], ['bugs']*len(df)), np.int32)
	# 랭킹 상승 평균값
	rankimean = np.fromiter(map(rankIncreaseMean, df['title'], ['bugs']*len(df)), np.int32)
	rankimean = np.array((rankimean+1) / (chartindays+1))
	# top5 진입 일수
	rankindays = np.fromiter(map(rankInDays, df['title'], ['bugs']*len(df)), np.int32)
	rankindays = np.array((rankimean + 1) / (chartindays+1))

	# 정규화
	chartindays = np.array((chartindays+0.01) / (chartindays+1))
	# 데이터 합치기
	test_data = np.array([chartindays, rankimean, rankindays]).T
	
	test_data
	
	x_data = test_data
	
	X = tf.placeholder(tf.float32)
	# 신경망에 가중치 W과 편향 b을 적용합니다
	L = tf.add(tf.matmul(X, W), b)
	
	L = tf.nn.sigmoid(L)
	
	model = L
	
	prediction = model
	result = sess.run(prediction, feed_dict={X: x_data})
	
	print('예측값:', result)
	
	df['prediction'] = result
	df
	
	saver = tf.train.Saver(tf.global_variables())
	saver.save(sess, './model/dnn.ckpt')

입력	질의문(테스트 음원의 수, 테스트 해당연월일)
출력	입력한 갯수의 테스트 음원들의 조작점수.