안녕하세요 10년간 전국 시군구 아파트 집값 상승률 분석을 주제로 진행한 빅데이터 프로그래밍

프로젝트 최종 발표를 하게 된 명지대학교 융합소프트웨어학부 권성중 입니다

다음과 같이 문제 정의 시스템 아키텍처 데이터수집방법

데이터분석방법 데이터 분석 결과 추가적인 확장 가능성 이렇게 6개로 목차를 나눠보았습니다.

첫번째 문제 정의입니다.

2020년 이후로 대한민국 뉴스에 가장 오르내리는 토픽은 부동산 문제였습니다. 2021년 현재에도 다양한 부동산 정책을 내놓았음에도 불구하고 문제는 해결되지 않았으며, 다주택자와 무주택자 간의 갈등 또한 점차 커지고 있습니다.

2021년 7월경 저는 현재 거주하고 있는 우리 동네가 2021년 상반기 집값 상승률 전국 1위라는 기사 제목을 우연찮게 포털사이트 메인에서 발견할 수 있었습니다.

10년 동안의 집값 상승률은 어느 시/군/구 지역이 높았는지, 각 연도 간에 어느 지역의 집값이 가장 많이 올랐는지, 10년 동안 통틀어 어느 지역의 집값이 평균적으로 비싸게 거래됐는지 궁금해져서 전국 아파트 실 거래가 데이터 셋을 바탕으로 한 빅데이터 분석을 하기로 결정했습니다.

둘째로 시스템 아키텍처 입니다.

SW스택으로는 python, spark를 툴로 사용하였고, 셀레니움, 판다스 이외에도 json, zipfile 등 다양한 모듈을 사용하며 프로젝트를 진행했습니다

데이터 분석을 위해 spark를 사용한 이유는

디스크 대신 메모리를 사용하는 스파크는 맵리듀스와 비교했을 때 최대 100배 빠른 속도를 보입니다. 많은 DB와 호환이 가능하며, scala 기반으로 작성된 것이 특징입니다.

빠른 속도와 현재 database 수업에서 배우고 있는Sql문으로 데이터 분석이 가능하다는 것이 가장 크게 다가와서 spark를 이용해 데이터분석을 하기로 결정하였습니다.

저장 방법 포맷으로는 csv를 사용하기로 결정했습니다.

Csv는 단순 텍스트 형식으로 서식 없이 데이터 값만으로 저장됩니다. 따라서 파일 용량이 적고 많은 양의 데이터를 축적할 때 주로 사용됩니다. 여러 프로그램으로 쉽게 파일을 확인할 수 있으며 데이터 관리하기에도 용이하기에 csv를 채택했습니다.

다음은 파이프라인인데 check year함수를 통해 1월1일인지 매일 체크하여 1월 1일일 경우 데이터 수집이 자동화 되도록 설계하였습니다.

크롤링 함수에서는 셀레니엄 모듈을 이용하여 행정동 표준 코드 관리시스템에 자동으로 접속하여 법정동 코드 zipfile을 저장하도록 설계하였습니다. API를 사용하려 했으나 법정동 코드에 대한 API가 유료로 존재하여 새로 구현하게 됐습니다.

Ziptotxt 함수를 통해 앞서 다운로드 받은 zip파일을 압축해제 및 txt파일명을 변경하였습니다.

Code\_extract함수를 통해 앞선 법정동 코드 txt 파일을 pandas dataframe으로 불러와 폐지된 지역 또한 도/특별시/광역시 등 API를 불러오는데에 필요 없는 행들을 제거해주었고 이렇게 전처리된 데이터프레임을 csv형태로 저장했습니다.

Collect데이터함수에 API를 불러오기 위한 인증키를 보관하고 call api함수를 실행후에 merge 함수를 실행하도록 구현 했습니다.

Call api함수에서는 앞서 전처리된 법정동 코드 csv파일을 한줄씩 불러와 API를 호출하였습니다. 얻은 데이터를 데이터프레임에 추가하고 지역간 정확한 집값 비교를 위해 제곱미터 당 가격을 컬럼으로 추가하여 거래가격 나누기 전용면적을 계산한 값과 함께 묶어주어 csv형태로 저장해주었습니다.

Merge\_csv 함수를 통해 call\_api 함수에서 한달씩 모은 데이터를 같은 년도끼리 merge해주었습니다

Merge\_csv\_all 함수를 통해 전체 데이터를 한곳으로 merge했습니다

앞서 생성된 빅데이터를 바탕으로 spark를 이용하여 데이터 분석을 진행했습니다.

셋째로 데이터 수집방법입니다.

기존의 시군구가 다른 시군구에 편입될 가능성을 고려해 Selenium을 이용하여 새로운 법정동 코드를 저장할 수 있도록 구현하였습니다.

안전한 저장을 위해 저장한 파일이 다운로드 디렉토리에 존재해야 드라이버가 종료되도록 했습니다.

앞서 다운로드 받은 zip 파일을 python파일이 존재하는 디렉토리에 압축해제시키고 얻은 txt파일 명을 변경해 주었습니다

API를 불러올 때에 법정동 코드는 상위 시/군/구를 나타내는 5자리만 필요합니다.

하지만 앞선 파일에서는 10자리로 동/면/읍 까지 나타내어 이를 전처리 하였습니다.

없어진 지역, API를 불러오지 못하는 도/특별시/광역시, 구를 포함하는 시를 제거해주고 전처리된 파일을 저장했습니다.

전국 한달치의 API를 불러오기 위해서는 250의 트래픽이 필요하지만 API에 허용된 트래픽이 일일 1000밖에 안되어 여러 서비스 키를 리스트에 저장해 두었습니다. Datetime 모듈을 이용하여 전년도를 자동으로 파악하여 매년 정상적으로 API를 불러올 수 있도록 구현했습니다. 앞선 서비스키와 연월 6자리을 call\_api 함수에 argument로 넣어주었고 1년치 데이터를 불러오면 merge함수를 실행하도록 하였습니다

데이터프레임을 만들기 위한 빈 리스트를 생성해 주었고 정제된 코드와 각 코드에 해당하는 시군구 명을 하나씩 불러오도록 구현했습니다.

API를 받아와 json형식으로 변환시켜주고 앞서 생성한 리스트에 값을 추가하여 이를 데이터프레임으로 만들어 주었습니다. 지역간 정확한 집값 비교를 위해 제곱미터 당 가격을 per\_price 컬럼으로 추가하여 거래가격 나누기 전용면적을 계산한 값과 함께 묶어주어 csv형태로 저장해주었습니다.

Glob모듈을 이용해 call\_api 함수에서 한달씩 모은 데이터를 같은 년도끼리 merge해주었습니다

Merge\_csv\_all 함수를 통해 전체 데이터를 한곳으로 merge했습니다

Datetime, time 모듈을 사용하여 하루에 한번 1월 1일인지 체크하고 1월 1일일 경우 main함수가 실행되면서 데이터 수집이 자동으로 진행되도록 구현했습니다. 이때에 시간 오차가 발생할 수 있기 때문에 하루-1초 씩 sleep상태로 두었습니다.

넷째로 데이터 분석 방법입니다.

앞서 얻은 빅데이터를 바탕으로 전처리한 데이터는 다음과 같습니다

월별로 수집한 데이터셋은 총 120개로 각 파일당 4~6만행, 각각 약 5mB의 용량입니다.

연별로 수집한 데이터셋은 총 10개로 각 파일당 40~60만행, 각각 약 50mB의 용량입니다.

전체 데이터셋은 약 524.5MB이고 약 602만행을 얻을 수 있었습니다.

데이터 분석을 위한 절차로는 향시적으로 home.py를 실행시키면 매년 1월1일마다 자동으로 데이터 수집을 하게 구현하였습니다.

Home.py가 위치한 디렉토리 속 yeardata폴더에 연도별 csv파일과 home.py가 위치한 디렉토리 속 merge.csvvㅏ일을 ambari hdfs 시스템에 옮깁니다.

SPARK 분석을 위해 짜인 코드인 allyear.py와 each.py를 발표자료 내 SW소스 코드 및 전략을 참고하여 필요에 따라 SQL문을 수정한 후 Cyberduck에 두 파일을 업로드 합니다.

Terminal을 열어 아래와 같은 명령어를 입력합니다.

Spark-submit 파일명.py를 하여 spark 데이터 분석을 실행합니다

업로드한 파이썬 파일 내에 설정된 경로에 분석 결과 csv파일을 찾은 후 다운로드 합니다.

SW 소스 코드 및 전략에서는 df1,df2에 분석하고 싶은 데이터셋 파일의 경로를 알맞게 수정하고 원하는 저장 경로를 지정합니다.

원하는 출력 행의 양을 SQL문 속 LIMIT 뒤 숫자 변경을 통해 선택할 수 있고 전체 행을 뽑고 싶은 경우 LIMIT을 포함하여 뒤에 숫자 까지 지우면 됩니다.

각각의 연도 별로 비교하기 위한 spark 코드 (eachyear.py)에서는

비교할 두 csv파일을 불러와 각각 region\_name(시/군/구)로 group by 하고 group 별로 average per\_price(제곱미터당 가격)을 구하여 t1,t2로 지정해 주었다. 그후 t1,t2의 region\_name이 같은 것 끼리 join하여 하나의 table로 나타내 주었습니다.

각 column을 설명하자면 t1.region\_name은 시/군/구, t1.p1은 df1의 제곱미터당 가격, t2.p2는 df2의 제곱미터당 가격, inc는 증가율을 나타냅니다.

내림차순으로 상위 10개 지역을 알아볼 수 있도록 구현하였다.

보고서에는 평당 가격이라 되있지만 이자리를 빌어 제곱미터 당 가격으로 정정합니다.

전체 기간 동안 연도별로 가장 비쌌던 지역을 확인하기 위한 코드 (allyear.py)에서는

전체 dataset을 가진 merge.csv를 불러와 year, region\_name 으로 group by 해주었다. 내림차순으로 상위 50개 지역을 알아볼 수 있도록 구현하였습니다.

앞서 봤던 것과 같이 전체 부분을 보고 싶다면 limit 이후 부분을 지워주면 됩니다.

pyspark.sql을 활용함으로서 개발 용이, 어떤 data를 분석한 것인지 쉽게 파악이 가능해진다. 일반인이 데이터 분석을 할 때에 변형 또한 SQL을 통해 쉽게 할 수 있도록 구현하였다

다섯째로 데이터 분석 결과입니다.

먼저 2012년 부터 2021년 까지 10년동안 집값 상승률을 보면 세종특별자치시가 가장 높았다.

과천에 위치한 정부청사를 비롯한 여럿 행정자치기구를 세종특별자치시로 옮기는 이슈를 바탕으로 시세가 많이 올랐다는 기사를 많이 보았는데 일치했다.

각 연도별로 1년 간 상승률을 비교했을 때, 대체로 지방에 위치한 집 값이 가장 많이 top 10에 올랐다.

수도권 보다 지방이 개발 이슈에 민감한 것과 동시에 수도권은 변동성이 적고 지방에 위치한 집값이 변동성이 큰 것도 한 몫 하는 것 같다.

2017년 가격과 2018년 가격을 상승률을 비교한 파일인데 실제로 2018년, 성동구에 많은 재개발 이슈와 재개발 건축이 진행되고 있었다.

전체 연도로 보았을 때 모두의 예상처럼 서초구, 강남구의 집 값이 가장 비싼 것을 알 수 있다.

정확한 집값은 몰랐지만 서초, 강남, 송파 지역이 비싸다고 알고 있었는데 가설과 일치했다.

마지막으로 추가 확장 가능성 입니다.

자동화를 통해 매년 데이터를 수집한다면 몇GB 수년이 흘러 TB 단위의 데이터를 수집할 수 있도록 구현하였다.

기본적인 SQL구문을 제공하고 이를 활용하여 사용자가 원하는 결과 값, 데이터 양을 얻을 수 있도록 구현했다. 추가로 사용자의 needs에 따라 원하는 결과를 구하기 위해 SQL을 입력하면 쉽게 data를 분석할 수 있도록 구현하였다.

예를 들어, merge.csv파일을 바탕으로 사용자가 원하는 특정 지역의 집값 추이를 확인할 수 있다.

ex) 강남구의 평당 집값 추이.