기초 빅데이터 프로그래밍 프로젝트 보고서

유종선
20191611

1. 개요 : 세상에는 정말로 많은 종류의 위스키들이 존재한다. 그래서, 어떤 위스키를 선택할지 고민 되는 경우가 많이 생긴다. 이때, 많은 사람들이 좋아하는 위스키는 어떤 것인지, 종류에 따라 어떤 위스키들을 사람들이 소비하는지에 대한 정보를 알고 싶은 경우가 있다. 이러한 정보를 프로젝트를 통해서 도출해낼 것이다. 위스키에는 특정한 정보가 존재한다. 얼마나 오랫동안 cask에 숙성시켰는지를 나타내는 연수, 종류, 도수, 위스키가 생산된 스코틀랜드 지역(위스키는 어떤 지역에서 생산되었는지 또한 맛과 향에 큰 영향을 미친다)이나 미국 지역 그리고 가장 중요한 가격이 있다. Input으로 들어오는 이러한 정보들을 분석하여 어떠한 연수가 혹은 도수가 맛있는 위스키일 확률이 높은지를 그래프를 통해서 나타낼 수 있다. 여기서 맛있다는 기준을, 현재 세계에서 가장 큰 위스키 평점 사이트인 www.whiskybase.com에서 추출하여 정보로 활용한다. 또한 여기서, 맛에 관한 정보들도 추출한다. 물론 맛이라는 것 자체가 주관적이지만, 여러 사람들이 느끼는 맛은 꽤 객관적인 정보로 활용될 수 있다. 그래서 맛을 추출하여, 이 맛을 바탕으로 위스키를 추천해주는 시스템 또한 같이 구현하였다.

2. 입력 : 입력파일은 txt 파일로 다음과 같이 구성되어있다.

```
Bowmore 12|12|Islay|81.92|40.0|Smoky|Citric|Vanilla|70000
Bowmore 15|15|Islay|83.95|43.0|Smoky|Sherried|Chocolate|145000
Bowmore 18 deep & complex|18|Islay|86.75|43.0|Smoky|Sherried|Chocolate|240000
Glendronach 12|12|Highlands|84.14|43.0|Sherried|Vanilla|Chocolate|100000
Glendronach 15 Revival|15|Highlands|86.34|46.0|Sherried|Chocolate|Honey|240000
Glendronach 18 Allardice|18|Highlands|88.57|46.0|Sherried|Chocolate|Old Wood|460000
Glendronach 21 Parliament|21|Highlands|89.29|48.0|Sherried|Chocolate|Dried Fruit|520000
Springbank 10|10|Campbeltown|86.90|46.0|Fresh Fruit|Citric|Honey|350000
```

앞에서부터 “이름|년수|지역|점수|도수|맛1|맛2|맛3|가격” 이러한 형태로 정보가 저장되어있다. 정보를 읽을 때는 당연히 “|” 기준으로 정보를 구별하여 저장한다.

3. 프로그램 순서
    1. 먼저 txt파일에서 위스키 리스트를 입력받는다. 입력 받은 whisky정보를 class에 저장한다.
    2. 저장이 끝나면, While 문으로 구성된 입력 창이 구동된다. 1 ~ 5까지 사용자는 5가지의 선택지를 가진다.
    3. 0을 입력받아 프로그램을 종료할 때까지 계속해서 구동된다.

4. 함수 설명

- 실행시간 측정 함수

```
def checktime(func): #실행시간
    def new_func(*args,**kwargs):
        start_time = time.time()
        result = func(*args,**kwargs)
        print("실행시간:",(time.time()-start_time))
        print("\n")
        return result
    return new_func
```

- Whisky_list라는 class에 각 정보들을 저장한다.

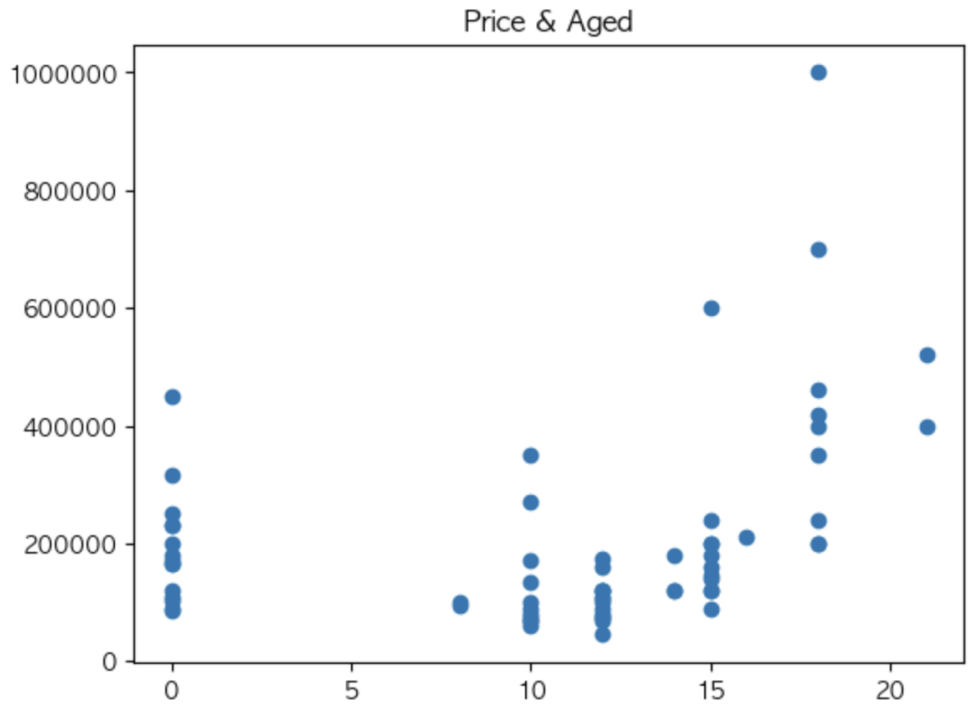
```
class Whisky_list():
    def __init__(self,name,aged,region,rating,strength,favor1,favor2,favor3,price):
        self.name = name          #이름
        self.aged = aged          #년수
        self.region = region      #지역
        self.rating = rating      #평점
        self.strength = strength #도수
        self.favor1 = favor1      #맛1
        self.favor2 = favor2      #맛2
        self.favor3 = favor3      #맛3
        self.price = price        #가격
```

- 입력으로 1을 입력받았을 경우(년수와 가격의 상관 관계) 코드와 결과값이다. 먼저 analysis_data함수로 보내어지고, K 값이 1이면 가격과 년수의 상관관계에 대한 그래프를 그려서 출력한다. 상관관계를 제대로 파악하기 위해 scatter를 사용하였다. 이를 사용하면 모든 위스키의 가격과 년수에 대해서 확인할 수 있기 때문이다.

```
@checktime
def analysis_data(data, k):
    x = list()
    y = list()
    tmp = 0
    if(k == 1): # 가격(Y)과 년수(X)
        for t in data:
            x.append(t.aged)
            y.append(t.price)

        plt.scatter(x,y)
        plt.gca().get_yaxis().get_major_formatter().set_scientific(False) #눈금을 자세하게 보기 위함
        plt.title("Price & Aged")
        plt.show()
        print("** 숙성년수 0은 NAS(숙성년수 미표기) 로 실제 0년 숙성이 아닌 숙성 년수를 표기하지 않은 위스키입니다. **")
```

다음은 결과 값이다. 아래에 보면 그래프와 실행시간을 확인할 수 있다. 그래프를 보면, 점들이 모여있는 구간이 년수가 증가할 수록 위치가 올라가는 것을 확인할 수 있다. 특히 10년과 20년 부근을 비교해보면 차이가 난다. 이는 년수가 증가함에 따라 가격도 증가한다는 것을 확인할 수 있다. 숙성년수 0으로 되어있는 NAS(숙성년수 미표기)의 경우에 숙성년수가 적힌 것과 다르게 범위가 생성되어 있는 모습을 확인할 수 있다. 이는 NAS의 경우 숙성년수가 적힌 것과는 다르게 봐야한다는 점을 알 수 있고, 동시에 일정 금액이상 올라가지 않는 점도 확인해볼 수 있다.



** 숙성년수 0은 NAS(숙성년수 미표기) 로 실제 0년 숙성이 아닌 숙성 년수를 표기하지 않은 위스키입니다. **
실행시간: 0.10080718994140625

- 위와 같은 analysis_data 함수에서 k 값이 2인 경우에 코드이다. 여기서는 가격과 평점의 상관 관계를 확인할 수 있다. 이것도 마찬가지로 scatter를 사용해서 표현했다.

```
elif(k == 2): # 가격(X)과 평점(Y)
    for t in data:
        x.append(t.price)
        y.append(t.rating)

    plt.scatter(x,y)
    plt.gca().get_xaxis().get_major_formatter().set_scientific(False) #눈금을 자세하게 보기 위함
    xticks = [100000, 200000, 400000, 600000, 800000, 1000000]
    plt.xticks(xticks)
    plt.title("Price & Ratings")
    plt.show()
```

결과 값을 보자. 그래프를 보면 가격대가 낮은 곳에서는 낮은 점수로 시작해서 점점 점수가 올라갈 수록 가격대가 올라가고 범위가 넓어지는 것을 확인할 수 있다. 이를 통해 높은 점수를 받는 위스키들은 대체로 가격이 비싸고, 그 범위도 넓음을 확인할 수 있다. 또, 낮은 점수를 받는 위스키들은 대체로 가격이 낮다는 것또한 학인할 수 있다.

실행시간: 0.10327506065368652

- 다음에는 위와 같은 analysis_data함수에서 k 가 3을 입력받았을 때의 경우이다. 보면 5가지 지역 중에서 하나를 선택받는다. 원하는 지역을 선택하면, 그 지역에서 생산된 위스키의 flavor를 전부 수집한다. 수집한 정보를 counter 함수를 사용하여 count하면, flavor가 몇 번 나왔는지를 전부 확인할 수 있다. 이번에는 빈도수가 주 데이터이기 때문에, pie chart를 활용하였다.

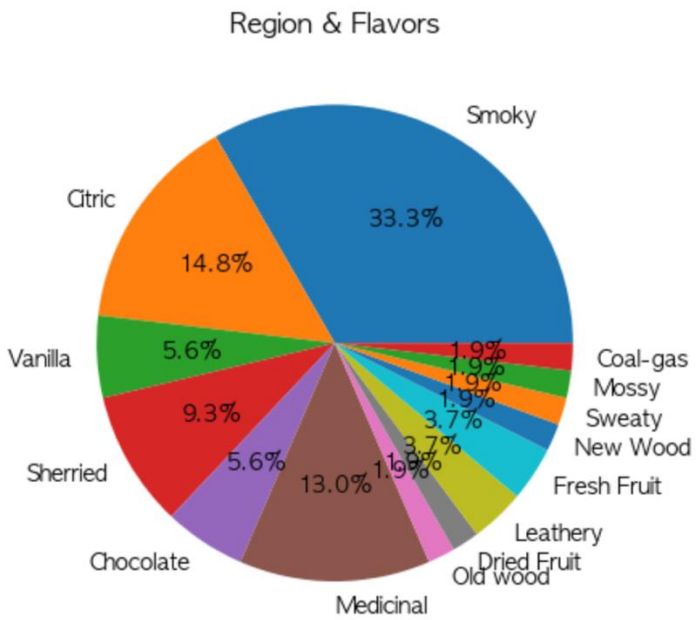
```
elif(k == 3):
    pie = list()
    flag = 0
    print("스코틀랜드에는 5가지 지역이 있습니다.")
    print("Islay, Campbeltown, Speyside, Highlands, Islands")
    word = input("원하시는 지역을 선택해 주세요 : ")
    for t in data:
        if(t.region == word):
            pie.append(t.favor1)
            pie.append(t.favor2)
            pie.append(t.favor3)
            flag = 1

    if(flag == 0):
        print("잘못된 입력입니다. ")
        return

    counts = Counter(pie)
    keys = counts.keys()
    values = counts.values()
    plt.title("Region & Flavors")
    plt.pie(values, labels = keys,autopct='%1.1f%%')
    plt.show()
```

결과를 보자. 보면 아래 예시는 Islay를 입력했을 경우이다. 보면 flavor에 따라서 비율이 나타나 있는 것을 확인할 수 있다. 이 지역의 위스키들은 대체로 어떤 맛을 많이 만들어내는지를 알 수 있다. Islay 지역의 경우 Smoky, Citric, Medicinal한 맛을 많이 만들어내는 것을 확인할 수 있다. 마지막에는 실행시간이 출력된다.

```
보고싶은 정보 넘버링을 입력해주세요 : 3
스코틀랜드에는 5가지 지역이 있습니다.
Islay, Campbeltown, Speyside, Highlands, Islands
원하시는 지역을 선택해 주세요 : Islay
```




```
실행시간: 3.336322069168091
```

- 이번에는 새로운 위스키 정보를 입력하거나, 위스키 이름으로 정보를 찾고 싶은 경우이다. 먼저 정보를 입력하고 싶으면 7을 입력한다. 그러면 아래와 같이 이름 ~ 가격까지 정보를 입력한다. 그리고 입력한 정보를 다시 파일에 써주고, 입력한 값을 return 해준다.
- 그리고 8을 입력하여 정보를 확인해보면 내가 입력한 값 그대로 잘 출력하고 있는 것을 확인할 수 있다.

```
@checktime
def input_data(data):
    while(1):
        try :
            name = input("이름 : ")
            aged = int(input("년수 :"))
            region = input("지역 : ")
            rating = float(input("점수 : "))
            proof = float(input("도수 : "))
            f1 = input("맛 1 : ")
            f2 = input("맛 2 : ")
            f3 = input("맛 3 : ")
            price = float(input("가격 : "))

            f = open("whisky.txt", "a")
            f.write("%s|%d|%s|%.2f|%.2f|%s|%s|%s|%d\n"%(name, aged, region, rating, proof, f1, f2,f3,price))
            f.close

            break


        except :
            print("잘못된 입력입니다. ")


    return name, aged, region, rating, proof, f1, f2, f3, price
```

```
보고싶은 정보 넘버링을 입력해주세요 : 4
위스키 정보 입력을 원하신다면 7을, 정보를 열람하고 싶으시다면 8를 입력해주세요 : 7
이름 : Auld Goonsy
년수 :12
지역 : Highlands
점수 : 86.8
도수 : 57.6
맛 1 : Sherried
맛 2 : Old Wood
맛 3 : Dried Fruit
가격 : 170000
저희는 약 70여종의 엔트리 위주의 위스키를 분석했습니다.
1 : 가격과 년수의 상관관계
2 : 가격과 평점의 상관관계
3 : 지역과 맛의 상관관계
4 : 새로운 위스키 정보 입력 or 출력
5 : 위스키 추천
만약 종료 하고 싶으시다면 0번을 눌러주세요.
보고싶은 정보 넘버링을 입력해주세요 : 4
위스키 정보 입력을 원하신다면 7을, 정보를 열람하고 싶으시다면 8를 입력해주세요 : 8
열람하고 싶으신 위스키의 이름을 영어로 입력해주세요(ex. Bowmore 12) : Auld Goonsy

name : Auld Goonsy, aged : 12년, region : Highlands, rating : 86.8,
strength : 57.6, favor : Sherried, Old Wood, Dried Fruit, price : 170000.0원

저희는 약 70여종의 엔트리 위주의 위스키를 분석했습니다.
1 : 가격과 년수의 상관관계
2 : 가격과 평점의 상관관계
3 : 지역과 맛의 상관관계
4 : 새로운 위스키 정보 입력 or 출력
5 : 위스키 추천
만약 종료 하고 싶으시다면 0번을 눌러주세요.
보고싶은 정보 넘버링을 입력해주세요 : 0
프로그램을 종료합니다. 감사합니다.
```

- 마지막으로 위스키 추천이다. 여기서는 여러가지 맛을 선택지로 준다. List를 만들어서 해당 맛이 들어 있는 위스키의 정보를 scored_list와 named_list에 저장해준다. 이 정보를 딕셔너리로 만든 다음, 점수를 기준으로 내림차순으로 정렬한다. Lambda 를 사용해 정렬을 하면 가장 앞의 위스키가 가장 높은 점수를 가지게 된다. 거기서 이름을 순서대로 다시 last에 추출하여 저장해둔다.
- 이중 for문을 돌면서, top3의 위스키들의 정보를 출력해준다.

```
@checktime
def recommend_whiskys(data):
    while(1):
        try :
            print("여기 여러가지 맛이 있습니다. ")
            print("당신이 원하는 맛을 선택하면, 그에 맞는 위스키를 점수(rating) 기반으로 추천해드립니다. ")
            print("Smoky : 피트향, 스모키함")
            print("Vanilla : 바닐라")
            print("Citric : 시트러스, 레몬, 라임향")
            print("Sherried : 셰리 와인")
            print("Chocolate : 초콜릿")
            print("Honey : 꿀")
            print("Fresh Fruit : 신선한 과일")
            print("Dried Fruit : 말린 과일 향")
            print("Toasted : 구운 빵")
            print("Old Wood : 오래된 나무 향")
            print("New Wood : 새 나무 향")
            print("Medicinal : 약품 향")
            print("Malt Extract : 몰트 향")
            a = input("가장 좋아하는 맛을 선택하세요 : ")

            scored_list = list()
            named_list = list()
            #scored = dict()
            for t in data:
                if(t.favor1 == a or t.favor2 == a or t.favor2 == a):
                    scored_list.append(t.rating)
                    named_list.append(t.name)


            scored = dict(zip(named_list, scored_list))
            sorted_dict = dict(sorted(scored.items(), key=lambda item: item[1], reverse = True))
            last = list(sorted_dict.items())


            print("\n3가지 위스키를 추천해드립니다. ")
            for x in range(0,3):
                for t in data:
                    if(t.name == last[x][0]):
                        print("%d. %s" %(x+1, t.name), end = "")
                        print("\t rating : %.2f, \tfavor : %s, %s, %s, \tprice = %d원" %(t.rating, t.favor1, t.favo

            break

        except :
            print("잘못된 입력입니다. ")
```

- 결과를 보면, 아래는 Medicinal을 입력한 경우이다. 차례대로 3가지 위스키를 추천해주었다. 위스키의 맛과 가격에 대한 정보도 포함되었다. 마지막에는 실행시간을 출력하였다.

```
보고싶은 정보 넘버링을 입력해주세요 : 5
여기 여러가지 맛이 있습니다.
당신이 원하는 맛을 선택하면, 그에 맞는 위스키를 점수(rating) 기반으로 추천해드립니다.
Smoky : 피트향, 스모키함
Vanilla : 바닐라
Citric : 시트러스, 레몬, 라임향
Sherried : 셰리 와인
Chocolate : 초콜릿
Honey : 꿀
Fresh Fruit : 신선한 과일
Dried Fruit : 말린 과일 향
Toasted : 구운 빵
Old Wood : 오래된 나무 향
New Wood : 새 나무 향
Medicinal : 약품 향
Malt Extract : 몰트 향
가장 좋아하는 맛을 선택하세요 : Medicinal

3가지 위스키를 추천해드립니다.
1. Lagavulin 16  rating : 87.03,        favor : Smoky, Medicinal, Old wood,     price = 210000원
2. Laphroaig Quater Cask         rating : 84.86,        favor : Smoky, Medicinal, Citric,       price = 90000원
3. Laphroaig 10  rating : 83.14,        favor : Smoky, Medicinal, Leathery,     price = 80000원
실행시간: 8.26561689376831
```

5. 프로젝트 보완 사항
   A. 위스키의 맛의 범위를 좁혔다. 작성자는 사람들이 많이 느끼는 맛을 위주로 선정하여 작성하였는데, 조금 더 많은 맛의 범위를 포함했으면 좋았을 것 같다.
   B. 데이터의 크기가 작았다. 대략 70여가지의 위스키들을 사용하여 진행하였는데, 이보다 훨씬 더 큰 정보를 활용하면 이보다 더 눈에 띄는 결과를 얻을 수도 있을 것 같다.
   C. 입력 받아야 하는 format이 정해져있어서 이를 반드시 지켜야 한다는 점을 보완했다면 사용자가 더 편리하게 사용했을 것 같다.