

패턴 인식 봄학기 프로젝트 과제

|  |  |
| --- | --- |
| **학 과 :** | **컴퓨터공학과** |
| **학 번 :** | **20171006** |
| **이 름 :** | **이찬영** |



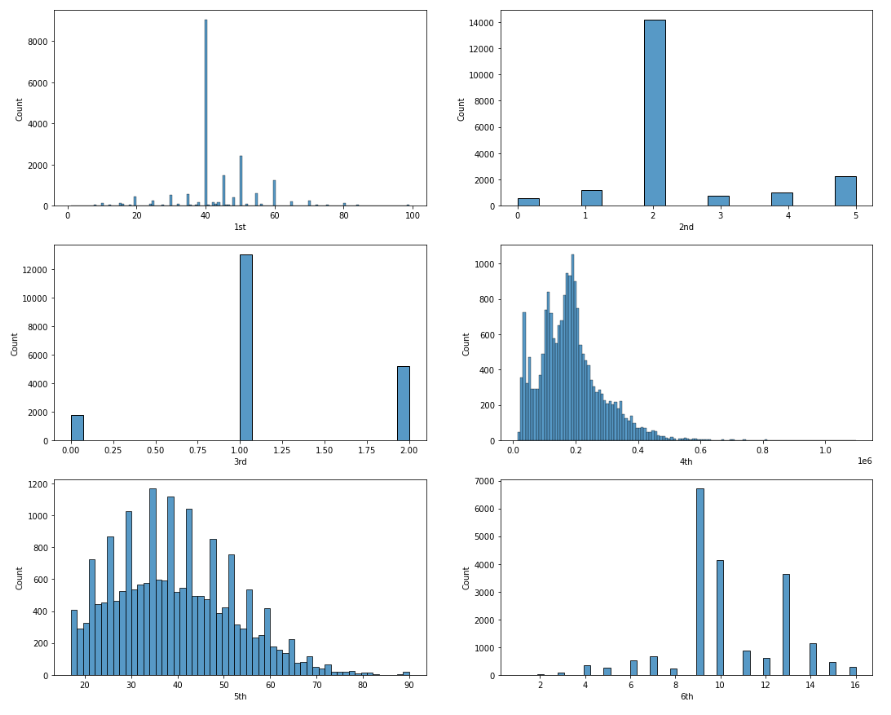
* KNN 분류기 설계

1. Null value 체크

: df.info()를 통해 data에 결측치가 있는지 체크한다.

1. Numeric, category col 분석

: Seaborn 라이브러리를 통해 column별로 히스토그램을 그려보며 category열과, numeric 열을 분리한다.



<column 별 histogram>

* 모든 column의 분포를 확인했을 때 1st, 4th, 5th, 6th 열은 numeric column이고, 2nd, 3rd 열은 category column임을 확인할 수 있다.

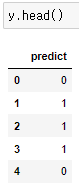
1. Data 분리, label data 숫자로 변경

: feature와 label을 x와 y로 분리한다.

* x = df.iloc[:, :-1] / y = df.iloc[:, 6:]

y = y.replace('unsatisfied',0).replace('satisfied',1)

* unsatisfied : 0 / satisfied : 1 변경

텍스트, 전자기기, 키보드이(가) 표시된 사진

자동 생성된 설명

1. One Hot Encoding & Standard Scaling

: category 열들은 연속적인 특성이 없는 데이터이기 때문에 연속성을 없애주기 위해서 One Hot Encoding을 적용합니다. Numeric 열들은 4th열과 5th열의 data의 경우와 같이 data의 scale이 크게 다르기 때문에 4th열의 중요도가 너무 높아질 우려가 있다. 따라서 standard scaling 작업을 통해 column의 범위를 같게 만들어줍니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<pipe\_processing 함수>

* Pipeline 라이브러리로 data를 one hot encoding, standard scaling 해줍니다.

1. KNN을 위한 함수 정의

5-1) euclidean\_distance 함수

* feature의 수가 n ( n > 2)개로 많기 때문에 모든 feature의 차를 제곱해서 더한 값의 제곱근을 구한다.
* 리스트에 모든 train 데이터와의 거리를 저장하고, 거리가 가까운 순으로 거리 리스트를 정렬한다.
* Numba 라이브러리 활용을 통해 python for문 속도 향상을 시킨다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<euclidean\_distance 함수>

5-2) get\_nearest\_neighbor 함수

* K로 설정한 값에 따라 k개의 가장 가까운 data의 라벨을 체크해서 k\_label에 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

<get\_nearest\_neighbor 함수>

5-3) get\_label 함수

* 리스트 내에서 최빈값을 예측값으로 결정한다.



<get\_label 함수>

5-4) knn 함수

* Test data의 수만큼 for문을 돌며 앞어 정의한 euclidean\_distance, get\_nearest\_neighbor, get\_label함수를 이용해 test data의 label을 예측한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

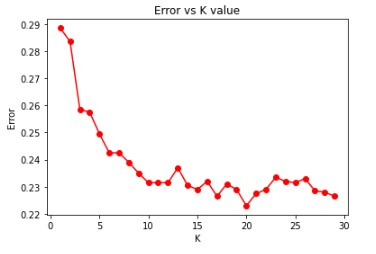
<knn 함수>

5-5) elbow 함수

* K 값을 1 ~ 30범위에 따라 변경해가며 error값을 계산한다.

K에 따른 error를 그래프를 통해 그려보고, 에러가 적은 K값을 K로 설정한다.

일정 K 이후로는 과적합의 우려가 있어서 K를 17로 설정하였다.



<k에 따른 error 값 그래프>

5-6) kfold 함수

* test\_size = 0.1로 프로젝트 조건에 따라 데이터 세트를 9:1로 나눈다.
* num\_folds = 10으로 설정하여 훈련/테스트 데이터를 10개 구성한다.
* 데이터 세트를 10등분 하고 9개는 학습 데이터로 1개는 테스트 데이터로 설정한다.

For문 반복을 통해 학습 데이터와 테스트 데이터를 변경해가며 예측을 진행한다.

y\_predict와 y\_test값 비교를 통해 정확도를 계산한다.

* 분류기의 성능은 10번 예측한 값의 평균을 통해 도출한다.

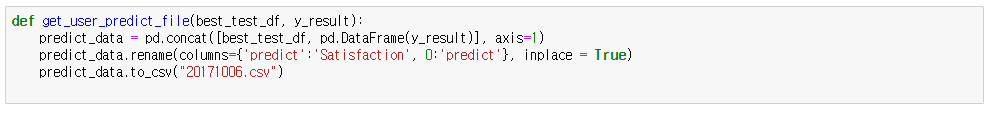
텍스트이(가) 표시된 사진

자동 생성된 설명

<kfold 함수>

5-7) get\_user\_predict\_file 함수

: kfold 결과를 진행할 때 가장 정확도가 높은 테스트 데이터의 응답자별 추정 만족도를 저장하기 위한 함수이다.



<get\_user\_predict\_file 함수>

테이블이(가) 표시된 사진

자동 생성된 설명

<응답자별 추정 만족도 데이터(20171006.csv)>

* Satisfaction은 원본 데이터의 만족/불만족 여부 값이고 predict 값은 knn모델을 통해 예측한 만족/불만족 여부 값이다.

(0 : 불만족, 1 : 만족)

* KNN 분류기 실행 시간 측정
* Datetime library를 이용하여 시간 측정

Kfold를 통해 10번 모델을 반복하는 시간을 측정하였다.



* KNN 분류기 결과

텍스트이(가) 표시된 사진

자동 생성된 설명

* 사용 언어

: 파이썬

* 소스코드 실행 환경 및 실행 방법
* 실행 환경 : jupyter notebook
* 실행 방법 : kernel -> Restart & Run All 클릭
* 실행 파일 생성 & 실행 방법
* 실행 파일 실행으로 knn 모델을 통해 분류를 할 수 있고, 훈련/테스트 데이터와 테스트 응답자 예측 결과 파일을 생성할 수 있다.

1. ipynb파일 .py파일로 변환

(그래프 plot하는 함수는 실행파일에 제외하였음.)

1. 클릭 (동일 경로에 satisfaction\_data.csv파일이 있어야 실행 가능)
2. py.exe 실행

텍스트이(가) 표시된 사진

자동 생성된 설명

<py.exe 실행 스크린샷>