데이터 마이닝

데이터 마이닝(data mining)은 대규모로 저장된 데이터 안에서 체계적이고 자동적으로 통계적 규칙이나 패턴을 찾아 내는 것이다. 다른 말로는 KDD(데이터베이스 속의 지식 발견, knowledge-discovery in databases)라고도 일컫는다.

데이터 마이닝은 통계학에서 패턴 인식에 이르는 다양한 계량 기법을 사용한다. 데이터 마이닝 기법은 통계학쪽에서 발전한 탐색적자료분석, 가설 검정, 다변량 분석, 시계열 분석, 일반선형모형 등의 방법론과 데이터베이스 쪽에서 발전한 OLAP (온라인 분석 처리:On-Line Analytic Processing), 인공지능 진영에서 발전한 SOM, 신경망, 전문가 시스템등의 기술적인 방법론이 쓰인다.

단점으로는, 자료에 의존하여 현상을 해석하고 개선하려고 하기 때문에 자료가 현실을 충분히 반영하지 못한 상태에서 정보를 추출한 모형을 개발할 경우 잘못된 모형을 구축하는 오류를 범할 수가 있다. [1]

적용 분야

- 분류(Classification): 일정한 집단에 대한 특정 정의를 통해 분류 및 구분을 추론한다 (예: 경쟁자에게로 이탈한 고객)
- 군집화(Clustering): 구체적인 특성을 공유하는 군집을 찾는다. 군집화는 미리 정의된 특성에 대한 정보를 가지지 않는다는 점에서 분류와 다르다 (예 : 유사 행동 집단의 구분)
- 연관성(Association): 동시에 발생한 사건간의 관계를 정의한다. (예: 장바구니안의 동시에 들어 가는 상품들의 관계 규명)
- 연속성(Sequencing): 특정 기간에 걸쳐 발생하는 관계를 규명한다. 기간의 특성을 제외하면 연관성 분석과 유사하다 (예: 슈퍼마켓과 금융상품 사용에 대한 반복 방문)
- 예측(Forecasting): 대용량 데이터집합내의 패턴을 기반으로 미래를 예측한다 (예: 수요예측)

회귀 분석(Regression)

통계학에서, 회귀분석(回歸分析, 영어: regression analysis)은 관찰된 연속형 변수들에 대해 두 변수 사이의 모형을 구한뒤 적합도를 측정해 내는 분석 방법이다.
회귀분석은 시간에 따라 변화하는 데이터나 어떤 영향, 가설적 실험, 인과 관계의 모델링등의 통계적 예측에 이용될 수 있다. 그러나 많은 경우 가정이 맞는지 아닌지 적절하게 밝혀지지 않은 채로 이용되어 그 결과가 오용되는 경우도 있다. 특히 통계 소프트웨어의 발달로 분석이 용이해져서 결과를 쉽게 얻을 수 있지만 적절한 분석 방법의 선택이였는지 또한 정확한 정보 분석인지 판단하는 것은 연구자에 달려 있다.

출처 : 위키피디아 (https://ko.wikipedia.org/wiki/%ED%9A%8C%EA%B7%80%EB%B6%84%EC%84%9D)

로지스틱 회귀 분석(Logistic Regression)

로지스틱 회귀(영어: logistic regression)는 D.R.Cox가 1958년[1] 에 제안한 확률 모델로서 독립 변수의 선형 결합을 이용하여 사건의 발생 가능성을 예측하는데 사용되는 통계 기법이다.
로지스틱 회귀의 목적은 일반적인 회귀 분석의 목표와 동일하게 종속 변수와 독립 변수간의 관계를 구체적인 함수로 나타내어 향후 예측 모델에 사용하는 것이다. 이는 독립 변수의 선형 결합으로 종속 변수를 설명한다는 관점에서는 선형 회귀 분석과 유

사하다. 하지만 로지스틱 회귀는 선형 회귀 분석과는 다르게 종속 변수가 범주형 데이터를 대상으로 하며 입력 데이터가 주어졌을 때 해당 데이터의 결과가 특정 분류로 나뉘기 때문에 일종의 분류 (classification) 기법으로도 볼 수 있다.
흔히 로지스틱 회귀는 종속변수가 이항형 문제(즉, 유효한 범주의 개수가 두개인 경우)를 지칭할 때 사용된다. 이외에, 두 개 이상의 범주를 가지는 문제가 대상인 경우엔 다항 로지스틱 회귀 (multinomial logistic regression) 또는 분화 로지스틱 회귀 (polytomous logistic regression)라고 하고 복수의 범주이면서 순서가 존재하면 서수 로지스틱 회귀 (ordinal logistic regression) 라고 한다.[2] 로지스틱 회귀 분석은 의료, 통신, 데이터마이닝과 같은 다양한 분야에서 분류 및 예측을 위한 모델로서 폭넓게 사용되고 있다.

출처 : 위키피디아(https://ko.wikipedia.org/wiki/%EB%A1%9C%EC%A7%80%EC%8A%A4%ED%8B%B1_%ED%9A%8C%EA%B7%80)

나이브 베이즈 분류

베이즈 정리를 기반으로 하는 단순한 확률 분류기

어떤 feature 변수 Fi가 주어졌을 때 이것이 Class C 에서 나왔을 확률인 사후확률을 측정하여 가장 확률이 높은 클래스로 분류

머신러닝 분야에서, '나이브 베이즈 분류(Naïve Bayes Classification)는 특성들 사이의 독립을 가정하는 베이즈 정리를 적용한 확률 분류기의 일종으로 1950 년대 이후 광범위하게 연구되고 있다.

통계 및 컴퓨터 과학 문헌에서 , 나이브 베이즈는 단순 베이즈, 독립 베이즈를 포함한 다양한 이름으로 알려져 있으며, 1960년대 초에 텍스트 검색 커뮤니티에 다른 이름으로 소개되기도 하였다.

나이브 베이즈 분류는 텍스트 분류에 사용됨으로써 문서를 여러 범주 (예: 스팸, 스포츠, 정치)중 하나로 판단한다.
또한, 적절한 전처리를 하면 더 진보 된 방법들 (예: 서포트 벡터 머신 (Support Vector Machine))과도 충분한 경쟁력을 보임을 알 수 있다.

예시 : 성별 분류하기, 스팸메일 분류하기

출처 : 위키피디아 (https://ko.wikipedia.org/wiki/%EB%82%98%EC%9D%B4%EB%B8%8C_%EB%B2%A0%EC%9D%B4%EC%A6%88_%EB%B6%84%EB%A5%98)

실습1 : 카테고리 예측기 만들기

코드 : category_predictor.py

#패키지 호출

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
```

#카테고리 맵 정의

```
category_map = {'talk.politics.misc': 'Politics', 'rec.autos': 'Autos',
        'rec.sport.hockey': 'Hockey', 'sci.electronics': 'Electronics',
        'sci.med': 'Medicine'}
```

#학습데이터셋을 받음

```
training_data = fetch_20newsgroups(subset='train',
        categories=category_map.keys(), shuffle=True, random_state=5)
```

 # countvectorizer 객체를 사용해 단어 빈도를 추출

```
count_vectorizer = CountVectorizer()
train_tc = count_vectorizer.fit_transform(training_data.data)
print("\nDimensions of training data:", train_tc.shape)
```

 # tf-idf 변환기 생성

```
tfidf = TfidfTransformer()
train_tfidf = tfidf.fit_transform(train_tc)
```

 # 테스트 데이터 정의

```
input_data = [
    'You need to be careful with cars when you are driving on slippery roads',
    'A lot of devices can be operated wirelessly',
    'Players need to be careful when they are close to goal posts',
    'Political debates help us understand the perspectives of both sides'
]
```

 # 다항 분포 나이브 베이즈 분류기 학습

```
classifier = MultinomialNB().fit(train_tfidf, training_data.target)
```

 # count vectorizer를 사용해 입력 데이터 변환

```
input_tc = count_vectorizer.transform(input_data)
```

 # tfidf 변환기를 사용해 벡터 데이터 변환

```
input_tfidf = tfidf.transform(input_tc)

# 카테고리 예측
predictions = classifier.predict(input_tfidf)

# 결과 출력
for sent, category in zip(input_data, predictions):
    print('\nInput:', sent, '\nPredicted category:', \
            category_map[training_data.target_names[category]])
```

NLTK를 이용한 감성 분석기 만들기

코드 : sentiment_alayzer.py

```
#패키지 호출
from nltk.corpus import movie_reviews
from nltk.classify import NaiveBayesClassifier
from nltk.classify.util import accuracy as nltk_accuracy

#입력 단어 리스트에서 자질 추출
def extract_features(words):
    return dict([(word, True) for word in words])

if __name__=='__main__':
    #말뭉치에서 리뷰를 로딩
    fileids_pos = movie_reviews.fileids('pos')
    fileids_neg = movie_reviews.fileids('neg')

#리뷰에서 특징 추출
    features_pos = [(extract_features(movie_reviews.words(
            fileids=[f])), 'Positive') for f in fileids_pos]
    features_neg = [(extract_features(movie_reviews.words(
            fileids=[f])), 'Negative') for f in fileids_neg]

    #학습셋과 데이터셋을 나눔 (80%:20%)
    threshold = 0.8
    num_pos = int(threshold * len(features_pos))
    num_neg = int(threshold * len(features_neg))

     #학습, 데스트셋 만들기
    features_train = features_pos[:num_pos] + features_neg[:num_neg]
    features_test = features_pos[num_pos:] + features_neg[num_neg:]

    #데이터 수 출력
    print('\nNumber of training datapoints:', len(features_train))
    print('Number of test datapoints:', len(features_test))

    #나이브 베이즈 분류기 학습
    classifier = NaiveBayesClassifier.train(features_train)
    print('\nAccuracy of the classifier:', nltk_accuracy(
            classifier, features_test))
```

```
#감정분석의 결정적인 단어 N개 출력
N = 15
print('\nTop ' + str(N) + ' most informative words:')
for i, item in enumerate(classifier.most_informative_features()):
    print(str(i+1) + '. ' + item[0])
    if i == N - 1:
        break
# 테스트에 사용할 샘플 문장 정의
# 영화 리뷰를 입력 데이터로 사용
input_reviews = [
    'The costumes in this movie were great',
    'I think the story was terrible and the characters were very weak',
    'People say that the director of the movie is amazing',
    'This is such an idiotic movie. I will not recommend it to anyone.'
]
# 샘플 데이터에 대해 예측 결과 출력
print("\nMovie review predictions:")
for review in input_reviews:
    print("\nReview:", review)

    # 확률 계산
    probabilities = classifier.prob_classify(extract_features(review.split()))

    # 가장 높은 값 선택
    predicted_sentiment = probabilities.max()

    # 결과 출력
    print("Predicted sentiment:", predicted_sentiment)
    print("Probability:", round(probabilities.prob(predicted_sentiment), 2))
```