삼성증권 SAMSUNG

2016. 1. 4

Quantitative Issue

기계학습(Machine Learning)과 투자전략

빅 데이터 기법: Naive Bayes Classifier의 활용

김동영, CFA
Analyst
dy76.kim@samsung.com
02 2020 7839

옥혜인
Research Associate
hyein.ok@samsung.com
02 2020 7795

빅 데이터 그리고 기계학습

빅 데이터는 최근 산업계의 가장 큰 화두다. 빅 데이터란 기존의 데이터 차원을 넘어서는 대량의 정형/비정형 데이터 집합 및 이런 데이터로부터 가치를 추출하고 결과를 분석하는 기술을 의미한다. 빅 데이터를 분석하는 작업은 인간의 처리 능력을 벗어나는 것이므로 기계의 도움이 필요하게 된다. 이 때 사용하는 도구가 바로 기계학습(Machine Learning) 알고리즘이다. 기계학습이란 컴퓨터가 데이터로부터 학습을 한 다음, 이를 바탕으로 데이터 분석 및 예측을 하는 알고리즘이다. 얼굴 자동인식 프로그램 혹은 스팸메일 자동분류 기능 등이 실생활에서 사용되는 기계학습의 사례에 해당한다. 미국 등의 선진국에서는 기계학습 기술을 활용하여 빅 데이터를 분석하는 'Data Scientist'의 수요가 폭발적으로 늘고 있는 상황이다.

기계학습(Machine Learning)은 주식투자 분야에서도 활용될 수 있다. 주식시장에는 수많은 데이터들이 존재한다. 매일매일 주식시장에 대한 뉴스 기사가 나오며, DART(전자공시시스템)에도 매순간 주식들의 공시정보가 올라온다. 각 주식의 여러 가지 펀더멘털 수치 데이터도 항상 조회할 수 있다. 이런 다양한 데이터들을 Machine Learning을 통해 자동 학습하여 패턴 인식 및 예측에 활용한다면, 자동화된 주식투자 모델을 만들 수 있다. 최근 증권업계에서 이야기되는 로보어드바이저 서비스에서도 기계학습 알고리즘이 일정부분의 역할을 하고 있다.

Naive Bayes Classifier 소개 및 모델 제시

나이브 베이즈 분류기는, 기계학습의 한 분야로서, 특정 자료가 여러 가지 속성을 가지고 있을 때 해당 자료를 어느 분류(Class)에 넣어야 할지를 베이즈 정리를 활용하여 판단해주는 알고리즘이다.

본고에서는 나이브 베이즈 분류기를 활용하는 W/L 모델과 BoW 모델을 제시한다. W/L 모델은 기존의 팩터분석 방법론과 유사하게 종목별 다양한 펀더멘털 지표를 기반으로 주가상승예상/주가하락예상 종목을 구분하는 모델이다. BoW 모델은 'Bag of Words' 방법을 사용하여 텍스트 마이닝을 할 수 있는 모델이다.

Naive Bayes Classifier 원리

$$P(C_k|x_1,\dots,x_n) \propto P(C_k)\prod_{i=1}^{n} P(x_i|C_k)$$

자료: 삼성증권

Contents

I. 빅 데이터 그리고 기계학습

빅 데이터는 최근 산업계의 가장 큰 화두다. 빅 데이터란 기존의 데이터 차원을 넘어서는 대량의 정형/비정형 데이터 집합 및 이런 데이터로부터 가치를 추출하고 결과를 분석하는 기술을 의미한다. 빅 데이터를 분석하는 작업은 인간의 처리 능력을 벗어나는 것이므로 기계의 도움이 필요하게 된다. 이 때 사용하는 도구가 바로 기계학습(Machine Learning) 알고리즘이다. 기계학습이란 컴퓨터가 데이터로부터 학습을 한 다음, 이를 바탕으로 데이터 분석 및 예측을 하는 알고리즘이다. 얼굴 자동인식 프로그램 혹은 스팸메일 자동분류 기능 등이 실생활에서 사용되는 기계학습의 사례에 해당한다. 미국 등의 선진국에서는 기계학습 기술을 활용하여 빅 데이터를 분석하는 'Data Scientist'의 수요가 폭발적으로 늘고 있는 상황이다.

기계학습(Machine Learning)은 주식투자 분야에서도 활용될 수 있다. 주식시장에는 수많은 데이터들이 존재한다. 매일매일 주식시장에 대한 뉴스 기사가 나오며, DART(전자공시시스템)에도 매순간 주식들의 공시정보가 올라온다. 각 주식의 여러 가지 펀더멘털 수치 데이터도 항상 조회할 수 있다. 이런 다양한 데이터들을 Machine Learning을 통해 자동 학습하여 패턴 인식 및 예측에 활용한다면, 자동화된 주식투자 모델을 만들 수 있다. 최근 증권업계에서 이야기되는 로보어드바이저 서비스에서도 기계학습 알고리즘이 일정부분의 역할을 하고 있다.

본고에서는 기계학습(Machine Learning) 모델의 일종인 Naive Bayesian Classifier에 대해서 소개하고, 해당 모델를 활용한 '기계의 힘을 빌린 주식투자전략'을 소개하고자 한다.

Contents

II. Naive Bayes Classifier 소개

베이즈 정리

Naive Bayes Classifier(나이브 베이즈 분류기)는 기계학습의 대표적인 알고리즘 중 하나다. 해당 알고리즘을 쓰기 위해서는, 우선 통계학에서 사용되는 '베이즈 정리'의 이해가 필요하다.

Bayes' theorem

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}$$

P(A)는 A의 확률, P(B)는 B의 확률
$P(A|B)$는 조건부 확률로, B가 주어졌을 때 A가 발생할 확률
$P(B|A)$는 조건부 확률로, A가 주어졌을 때 B가 발생할 확률

베이즈 정리는 위와 같다. $P(A|B)$의 확률을 $P(B|A)$, $P(A)$, $P(B)$를 통해서 계산하는 공식이다.

$$P(A|B)P(B) = P(A,B) = P(A)P(B|A)$$ ($P(A,B)$는 A 와 B 가 동시에 발생할 확률)

위의 식이 당연히 성립하므로, 베이즈 정리도 실상 당연한 팩트를 표현한 것일 수 있다. 하지만, 베이즈 정리는 '*P(A|B)*의 확률을 *P(B|A)*의 확률을 이용해서 계산한다'는 강력한 장점을 가지고 있다. 아래의 예를 살펴보자.

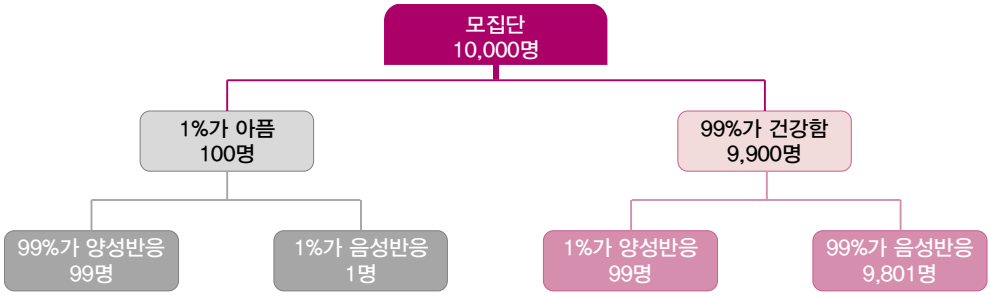
문제 예제)

특정 A질병에 대해서, 임상적으로 아래의 통계가 확인된다.
조건1) 전체 인구의 1%만이 이 질병에 걸린다(해당 질병으로 아프다).
조건2) 아픈 사람의 99%가 A질병 검사에서 양성반응을 보인다.
조건3) 건강한 사람의 99%가 A질병 검사에서 음성반응을 보인다.
질문) 특정인이 검사에서 양성반응을 보였다. 이 사람이 정말 A질병에 걸렸을 확률은 얼마인가?

위 질문에, 대략 눈으로 훑어봐서는 정답을 맞추기가 쉽지 않을 수 있다.
(왠지 정답은 99%나 1%가 아닐까 하는 생각이 먼저 들 수도)

위 질문의 답에 대해, 직관적인 해석을 위한 트리 다이어그램을 도입해보면 다음과 같다.

트리 다이어그램



자료: 삼성증권

10,000명의 모집단이 있다고 생각해보자. 위 그래프를 보면, 실제로 아픈 사람 중에서 양성반응을 보인 사람이 99명으로 나타난다. 건강한 사람 중에서 양성반응을 보인 사람도 99명으로 나타난다. 따라서 어떤 사람의 검사 결과가 양성이라면, 이 사람은 실제로는 건강할 확률과 아플 확률이 모두 50%로 나타난다. (99/(99+99)=50%)

베이즈 정리를 이용하면, 질문의 답을 아래와 같이 쉽게 계산할 수 있다.

조건1) $P(\text{아픔}) = 1\%$
조건2) $P(\text{양성}|\text{아픔}) = 99\%$
조건3) $P(\text{음성}|\text{건강}) = 99\%$
질문) $P(\text{아픔}|\text{양성}) = ?$

$$P(\text{아픔}|\text{양성}) = \frac{P(\text{아픔})P(\text{양성}|\text{아픔})}{P(\text{양성})} = \frac{P(\text{아픔})P(\text{양성}|\text{아픔})}{P(\text{아픔})P(\text{양성}|\text{아픔}) + P(\text{건강})P(\text{양성}|\text{건강})}$$

$$= \frac{0.01 \times 0.99}{0.01 \times 0.99 + 0.99 \times 0.01} = 0.5 = 50\%$$

나이브 베이즈 분류기

나이브 베이즈 분류기는, 특정 자료가 여러 가지 속성을 가지고 있을 때 해당 자료를 어느 분류(Class)에 넣어야 할지를 베이즈 정리를 활용하여 판단해주는 기계학습 알고리즘이다.

예를 들어, 어떤 사람의 키, 몸무게, 허리둘레 정보만을 알고 있는 상황에서 이 사람이 남성인지 여성인지 구별하기 위해서 나이브 베이즈 분류기를 사용할 수 있다. (분류기는 과거의 많은 남성의 통계와 여성의 통계를 참조해서 분류 작업을 진행하게 된다) 어떤 주식의 실적 증가율, 밸류에이션 정보 등을 넣었을 때, 해당 종목이 통계적으로 주가 상승 종목인지, 하락 종목인지 분류하는데도 해당 분류기가 사용될 수 있다.

나이브 베이즈 분류기는, n개의 속성을 의미하는 벡터 $X = (x_1, \dots, x_n)$가 주어졌을 때, 특정 클래스 C_k일 확률을 베이즈정리를 통해 계산하여 가장 확률이 높은 클래스를 선택하게 된다.

$X = (x_1, \dots, x_n)$의 속성이 주어졌을 때, 해당자료가 C_k에 속할 확률은 아래와 같이 표현된다.

$$P(C_k|X) = P(C_k|x_1, \dots, x_n)$$

$$P(C_k|X) = \frac{P(C_k)P(X|C_k)}{P(X)}$$

위의 식을 다음과 같이 지칭하기도 한다 $posterior = \frac{prior \times likelihood}{evidence}$

(posterior=사후, prior=사전, likelihood=공산, evidence=관찰 값)

$$P(C_k|x_1, \dots, x_n) = \frac{P(C_k)P(x_1, \dots, x_n|C_k)}{P(X)}$$

나이브 베이즈 분류기 명칭에 '나이브'가 들어간 이유는 $(x_1, \dots, x_2)$의 x_i가 모두 서로 독립이라고 러프하게 가정하기 때문이다. 독립이라는 가정을 사용하면 다음처럼 식을 변경할 수 있다.

$$P(C_k|x_1, \dots, x_n) = \frac{P(C_k)P(x_1, \dots, x_n|C_k)}{P(X)} = \frac{P(C_k)P(x_1|C_k)P(x_2|C_k)P(x_3|C_k)\dots}{P(X)}$$

$$P(C_k|x_1, \dots, x_n) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i|C_k)}{P(X)}$$

$P(X)$는 이미 발생한 결과이므로 C_k와는 무관한 상수다. $P(X)$를 제외한 비례식은 다음과 같다.

$$\boldsymbol{P(C_k|x_1, \dots, x_n) \propto P(C_k)\prod_{i=1}^{n} P(x_i|C_k)}$$

위 식이 나이브 베이즈 분류기에서 최종적으로 사용하는 산식이다.

나이브 베이즈 분류기에서 각 클래스(C_k)별 속성의 확률 분포는, training data 를 통해서 학습된다. 클래스의 정보가 학습된 다음, 분류하고자 하는 '한 사건'의 $(x_1, \dots, x_n)$ 속성이 주어지면, 각 클래스(C_k)별 속성의 확률 분포와 같이 비교하여, 이를 통해 해당 사건이 어떤 클래스에 더 가까운지를 확률값으로 계산할 수 있게 된다.

모델의 구체적인 사용 방법은 다음에 나오는 샘플 모델을 통해서 설명하도록 한다.

Contents

III. Bayes 1: 가칭 W/L 모델

W/L 모델 소개

여기서는 주식투자전략에 활용할 수 있는 나이브 베이즈 모델인 가칭 Winner/Loser 모델을 제시하고자 한다.

팩터분석을 비롯한 퀀트전략 대부분의 기본 원리는 꾸준하게 주가 상승으로 이어지는, 혹은 주가 상승과 연관되는 투자지표들을 찾으려는 '패턴 분석'의 과정이다. 즉, 과거에 주가 상승과 밀접했던 투자지표를 찾아서, 앞으로도 해당 지표로 투자하면 초과성과를 낼 것이라는 것이 기본적인 접근법이다.

이와 비슷하게, 과거에 항상 KOSPI를 아웃퍼폼했던 승자(Winner) 종목군과 언더퍼폼했던 패자(Loser) 종목군을 나눠서 각각의 특징을 정리한 다음, 개별 종목 투자 시 Winner군에 가까운지 Loser군에 가까운지를 비교하여 투자판단을 하는 모델을 생각해 볼 수 있다.

Winner/Loser 분류를 위한 사전 정보 (Training data)

1. KOSPI200종목의 매 분기별 펀더멘털 정보, 12분기치 (2012년 9월말~2015년 6월말 기간. 샘플개수=200×12=2400개)
2. 각 샘플의 클래스(Winner, Loser) 분류기준: 주가의 다음 분기 상대수익률이 0% 이상이면 Winner, 다음 분기 상대수익률이 0% 미만이면 Loser로 분류
3. 속성(펀더멘털) 세부 데이터: 분기말 기준 Book to Price (P/B의 역수), 매출액 증가율, 목표주가 괴리도, 주가 변동성, ROE → 총 5개의 속성을 사용함

주가 Winner 그룹과 Loser 그룹의 특징을 구분하기 위해, KOSPI200 종목의 12분기 데이터 총 2400개의 표본을 training data로 사용한다.

Training data의 일부 샘플을 표시하면 다음과 같다.

Winner/Loser 클래스를 위한 training data 샘플

W/L클래스	다음분기 상대수익률 (%)	속성1 B/P (%)	속성2 매출액증가율 (%)	속성3 TP괴리도 (%)	속성4 주가변동성 (%)	속성5 ROE (%)	비고
Winner종목	13.0	63.4	20.1	27.4	30.0	20.1	삼성전자, 12년 12월 시점
Winner종목	20.9	70.7	43.3	29.3	32.3	26.2	SK하이닉스, 13년 12월 시점
Winner종목	12.4	220.0	1.3	31.8	26.3	16.8	한국전력, 15년 9월 시점
Loser종목	−13.3	75.3	8.8	23.8	33.8	21.1	현대차, 12년 12월 시점
Loser종목	−7.3	66.7	15.0	20.7	35.9	22.9	현대모비스, 12년 12월 시점
Loser종목	−5.2	92.7	1.8	36.7	22.6	13.4	삼성전자, 15년 9월 시점
⋮							

참고: 총자료수는 2400개임
자료: 삼성증권

2400개의 표본을 가지고, Winner(상대주가 상승종목)와 Loser(하락종목)의 특성(평균값, 표준편차)를 계산해낼 수 있다. 이 때 매출액 증가율, Book to Price 수치 등은 기본적으로 Outlier 값이 많이 들어 갈 수 있다. 따라서 Outlier 5%에 해당하는 값을 경계값으로 바꿔버리는 Winsorization 기법을 통계 산출 시에 사용했다.

2400개 표본을 가지고 최종적으로 확인된 Winner 클래스와 Loser 클래스의 특성은 다음과 같이 나온다.

클래스별 정보 (특성)

클래스	항목	Book to Price (%)	매출액 증가율 (%)	TP괴리율 (%)	주가변동성 (%)	ROE (%)
Winner	평균	98.9	8.6	26.8	33.4	10.0
	표준편차	50.2	11.7	12.0	7.4	6.3
Loser	평균	96.5	7.8	26.5	33.9	10.1
	표준편차	47.4	10.6	11.3	7.6	6.2

참고: 5%의 Winsorization을 적용한 결과임
자료: 삼성증권

각 수치에 대해 잠깐 확인해보자. Winner 클래스, 즉 주가상승 종목들의 Book to Price 평균값은 98.9%이고, Loser 클래스의 평균값은 96.5%로 나온다. 주가상승 종목들의 P/B 밸류에이션이 주가하락종목들보다 평균적으로 약간 더 낮았음을 알 수 있다. 매출액 증가율 수치도 Winner 클래스가 8.6%로 좀 더 높고, Loser 클래스가 7.8%로 좀 더 낮다. 이는 밸류에이션이 낮고, 실적 증가율이 높으면 주가가 상승하는 경우가 많다는 것을 의미한다. 일반적인 상식과 부합하는 결과다. ROE 수치를 보면, Winner는 평균 10.0%, Loser는 10.1%로 나온다. 이 결과는 고ROE가 주가 상승으로 연결되지 않았으며, 주가상승 종목과 주가하락 종목의 ROE 차이가 별로 없었다는 사실을 알려준다 (KOSPI200 종목의 최근 3년간의 움직임을 대상으로 했을 때).

이제, 나이브 베이즈 분류기를 본격적으로 적용할 차례다. 현재 특정종목의 속성 정보(Book to Price 값, 매출액 증가율 값 등)을 알고 있다면, 분류기를 통해 해당 종목이 Winner에 가까운지 Loser에 가까운지 판단할 수 있다. 즉 분류기의 결과를 가지고 해당 종목을 Buy할지 Sell할지 결정할 수 있는 것이다.

Winner/Loser 판단이 필요한 종목 샘플

샘플	Book to Price (%)	매출액 증가율 (%)	TP괴리율 (%)	주가변동성 (%)	ROE (%)
A종목	217.9	0.7	31.5	25.5	9.0

자료: 삼성증권

A종목이 Winner에 가까운지 Loser에 가까운지 판단해보자. 여기서 클래스 각 속성의 확률 분포는 정규 분포라고 가정한다. 정규 분포에서 사용되는 확률밀도함수 공식은 다음과 같다. 연속확률변수를 가정하였으므로, 확률수치에는 확률밀도 값을 사용한다.

$$\text{정규분포의 확률밀도함수 (probability density function)} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Book to Price=217.9%, 매출액증가율=0.7% 등등의 속성을 가진 A종목이 Winner에 속할까? Loser에 속할까? $P(Winner|x_1,\dots,x_n)$와 $P(Loser|x_1,\dots,x_n)$의 확률값을 비교하면 판단할 수 있다. 클래스별 정보 테이블에서 계산한 각 속성별 확률분포를 활용하여, A종목이 각 클래스에 들어갈 확률값을 아래와 같이 산출할 수 있다.

P(A종목이 Winner일 확률) 계산

$$(Winner|Book\ to\ Price, \dots, ROE)$$

$$\propto P(Winner) \times P(Book\ to\ Price|Winner) \times \dots \times P(ROE|Winner)$$

이때, $P(Winner) = 0.5$로 지정 (특정종목을 뽑았을 때, Winner일지 Loser일지에 대한 사전적인 확률. 특별히 사전정보가 없다고 하면 0.5로 두면 된다. 참고로 2400개 샘플에서는 Winner가 1202개, Loser가 1198개 있었음)

$$P(Book\ to\ Price|Winner) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi \times 50.2^2}} e^{\frac{-(217.9-98.9)^2}{2\times 50.2^2}} = 0.00048$$

(Winner의 Book to Price 속성을 training data로 훈련한 결과, Winner의 B/P는 평균값이 98.9%, 표준편차가 50.2%로 나왔음(7페이지 표). 이때 B/P=217.9%인 Winner 종목의 pdf 수치는 위와 같이 계산됨)

$$P(ROE|Winner) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi \times 6.3^2}} e^{\frac{-(9.0-10.0)^2}{2\times 6.3^2}} = 0.0629$$

동일한 방법으로 $P(\text{매출액증가율}|Winner)$, $P(TP\text{괴리율}|Winner)$, $P(\text{주가변동성}|Winner)$를 계산할 수 있다. 그 다음으로는 확률의 곱셈만 계산하면 된다.

이 때, 낮은 확률의 연속적인 곱셈은 소수점 단위의 정확도에 문제가 생길 수 있다. 따라서 나이브 베이즈 분류기에서는 확률값을 보통 log수치로 변화하여 계산한다.

$$Winner\text{일 확률수준} = \log(P(Winner|X)) = \log(0.5 \times 0.00048 \times \dots \times 0.0629) = -21.7$$

P(A종목이 Loser일 확률) 계산

$$(Loser|Book\ to\ Price, \dots, ROE)$$

$$\propto P(Loser) \times P(Book\ to\ Price|Loser) \times \dots \times P(ROE|Loser)$$

$P(Loser) = 0.5$로 지정

$$P(Book\ to\ Price|Loser) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi \times 47.4^2}} e^{\frac{-(217.9-96.5)^2}{2\times 47.4^2}} = 0.00032$$

$$P(ROE|Loser) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi \times 6.2^2}} e^{\frac{-(9.0-10.1)^2}{2\times 6.2^2}} = 0.0633$$

동일한 방법으로 $P(\text{매출액증가율}|Loser)$, $P(TP\text{괴리율}|Loser)$, $P(\text{주가변동성}|Loser)$를 계산할 수 있다.

$$Loser\text{일 확률수준} = \log(P(Loser|X)) = \log(0.5 \times 0.00032 \times \dots \times 0.0633) = -22.0$$

여기서, A종목의 P(Book to Price|Winner) 와 P(Book to Price|Loser) 수치를 확인해 보자. Winner클래스의 B/P 평균값은 98.9%, Loser클래스의 B/P 평균값은 96.5%이다. A종목의 B/P 값 217.9%는 Winner 평균과 Loser 평균에서 둘다 멀리 있는 값이지만, Winner 그룹에 좀더 가깝고 Loser 그룹과는 더 멀다. 따라서, P(Book to Price|Winner) 의 pdf값(0.00048)이 P(Book to Price|Loser)의 pdf값(0.00032)보다 크게 나오는 것이 당연하다.

나이브 베이즈 분류기로 돌려본 결과, A종목의 $\log(P(Winner|X))$는 -21.7, $\log(P(Loser|X))$는 -22.0으로 나왔다. A종목이 가진 정보인 Book to Price 값, 매출액증가율 값 등 5개 펀더멘털 수치를 가지고 Winner 그룹 및 Loser 그룹의 특성과 비교해보니, Winner일 확률값이 더 높은(Winner 클래스와 더 유사한) 것으로 나왔다. 이에 따라 A종목은 Winner 클래스로 분류할 수 있다.

이제, 여러 종목들에 대해 위의 과정을 동일하게 진행하여, Winner/Loser를 분류할 수 있다. 이 결과에 따라 매수종목, 매도종목으로 선택하는 투자전략을 사용할 수 있게 되었다.

예를 위해, 시가총액 상위 종목들을 현재 펀더멘털 수치를 활용하여 W/L모델로 분류한 결과는 다음과 같이 확인된다.

시가총액 상위 종목, W/L 모델 결과

종목명	Book to Price (%)	매출액 증가율 (%)	TP괴리율 (%)	주가변동성 (%)	ROE (%)	최종 클래스
삼성전자	94.0	1.6	31.2	24.2	11.4	Loser
현대차	165.8	3.5	33.3	32.9	10.6	Loser
한국전력	223.4	0.6	34.9	25.3	9.1	Winner
삼성물산	66.6	70.4	49.7	52.0	2.8	Winner
현대모비스	121.3	5.8	25.8	25.8	12.4	Loser

참고: 여기서 Winner, Loser 기준은 향후 3개월간 주가수익률이 KOSPI를 아웃퍼폼 하는지를 기준으로 삼음
계량적 분석에 의한 판단이며, 당사 기업분석 투자의견과 다를 수 있음
자료: 삼성증권

W/L 모델의 장점

이상으로 주식의 몇가지 펀더멘털 정보와 나이브 베이즈 분류기 알고리즘을 사용하는 W/L모델을 만들어 봤다. 과거에 주가가 상승한 종목들을 Winner 그룹으로 묶고 하락한 종목들을 Loser 그룹으로 묶은 다음, 현재 투자할 종목의 펀더멘털 수치를 가지고 Winner에 속할지 Loser에 속할지 판단하여 투자판단을 내리는 구조다. 이는 팩터분석 모델(과거에 주가 상승과 연관성이 높았던 투자지표를 추출한 다음, 해당 투자지표로 스크리닝하여 종목을 찾는 방식)과 큰 틀에서 봤을 때 유사한 접근법이다.

그러나, 나이브 베이즈 분류기를 사용할 경우 몇 가지 추가적인 장점을 가질 수 있다.

첫째, 변별력이 약한 투자지표를 스스로 걸러내는 기능이 있다. 기존의 팩터분석 모델에서는, 최종적인 종목 스크리닝에 쓰일 투자지표를 사람이 선택하게 된다. 수동으로 투자지표를 선택하는 과정에서 변별력이 높은 지표를 쓰면 전략이 잘 작동하겠지만, 변별력이 낮은 지표를 쓰면 전략이 실패할 수 있다. 기계가 특별히 개입할 수 있는 부분이 없다. 그러나, 나이브 베이즈 분류기에서는 수많은 속성 정보를 Input으로 다 넣더라도, 유용한 속성 정보에 따라서만 판단이 되도록 자동으로 조정하는 기능을 가지고 있다. W/L 모델에서 현재의 training data에 의해 훈련을 시켜보면 Winner클래스의 ROE 평균값과 Loser클래스의 ROE 평균값이 거의 유사하게 나타난다. 따라서 ROE 지표는 상승종목과 하락종목을 구별하는데 큰 변별력이 없는 지표라 할 수 있다. (정확히는 표준편차도 감안해야 함) 분류기를 통해서, 특정종목의 Winner/Loser 분류를 할 때 ROE에 의한 확률값은 항상 비슷하게 나오므로, 분류작업에서 거의 중립적인 역할을 하게 된다. 속성 정보의 경/중 자체를 training 과정에서 자동적으로 판단하여 반영하는 것이 나이브 베이즈

분류기의 특징에 해당한다. 예를 들어 W/L모델에서 쓰는 종목별 펀더멘털 데이터를 지금까지의 5개 정보가 아닌 100개 이상의 정보로 확대할 경우에도, Winner/Loser 모델에서는 알아서 중요한 정보에 기초하여 분류 작업이 진행된다. 이 부분이 나이브 베이즈 분류기와 같은 기계학습 알고리즘의 일반적인 장점이라 볼 수 있다.

둘째, 전략의 점진적인 자동교체가 가능하다. 기존의 퀀트전략 방식인 팩터분석 모델을 예로 들면, 해당 모델의 운영에는 인간의 판단 혹은 수작업적인 측면이 들어간다. 투자전략을 가끔씩 업데이트한다고 할 때, 어떤 지표들을 어떤 가중치로 결합할지에 대해 그때그때 주관적인 판단이 들어간다. Trend following 형태의 전략을 취한다고 할 때에도, 자동으로 최적의 멀티팩터 전략을 탐색하도록 자동화하기는 다소 어렵다. 하지만, 나이브 베이즈 분류기는 종목투자 전략의 연속적인 교체가 쉽다. W/L모델을 기준으로 해서, "KOSPI200종목의 항상 직전 12개 분기(3년치) 데이터를 training data로 쓰는" 모델을 구성했다고 해보자. 항상 직전 12개 분기 데이터를 쓰도록 모델을 구성해 놓으면, 매 분기가 지날 때 마다 W/L모델 안의 Winner클래스 확률분포와 Loser클래스 확률분포는 항상 업데이트된다. 항상 최근 3년치의 유효했던 스타일을 확인하고 그 스타일대로 향후 투자종목을 찾는 과정이 끊김 없이 자동적으로 이어지게 된다. 항상 새로운 정보에 맞춰 모델의 내부 변수를 자동으로 학습시키는 것이 이런 기계학습 알고리즘 만의 장점이다.

W/L 모델의 단점

W/L 모델의 단점은, 클래스 분류 작업은 쉬우나 강약의 정도를 측정하기가 어렵다는 점이다. 분류 위주의 알고리즘이므로 상승(예상)종목 혹은 하락(예상)종목 등으로의 분류는 쉽지만, 어느 정도의 주가상승률을 보일지, 같은 Winner클래스 중에서 상대적으로 어느 수준인지를 그 정도를 측정하기는 약간 어렵다. 단, Winner/Loser의 이원분류가 아닌 최상/상/중/하/최하 등으로 Class를 좀 더 세분화하는 것은 충분히 가능하므로, Class 세분화를 통해서 이런 단점을 일부 극복할 수 있다.

Contents

Ⅳ. Bayes 2: BoW (Bag of Words) 모델

BoW (Bag of Words) 모델 소개

사실, 나이브 베이즈 분류기가 가장 많이 사용되는 분야는 Bag of Words (BoW) 모델 분야다. BoW 모델은 문서(텍스트) 데이터를 처리하는 모델로서, 스팸메일의 자동필터, 텍스트의 감정 파악 등과 같은 '텍스트 마이닝' 분야에서 활발히 사용되고 있는 모델이다.

BoW 모델은 텍스트를 단어의 집합으로 표현하고 관리한다. 일련의 텍스트 데이터가 있을 때 이것을 BoW 자료형으로 바꾼다고 하면, BoW 자료에는 텍스트 안에 있었던 단어들이 이런이런 것들이 있고, 어떤 단어가 몇 번 존재했다는 정보가 들어가기 된다.

나이브 베이즈 BoW 모델은, 서로 다른 클래스에 해당하는 training text들을 사전에 훈련시킨 다음, 새로운 testing text가 주어졌을 때 해당 text가 어떤 클래스에 들어가는 지를 판단하는 알고리즘이다.

해당 모델을 통해, 스팸메일과 비스팸메일의 training text를 입력시켜 훈련시킨다고 하면, 새로운 메일이 오면 이게 스팸메일인지 비스팸메일인지 자동으로 판단할 수 있게 된다. 또한, 긍정적인 감정 표현의 training text와 부정적인 감정 표현의 training text를 훈련시킨 다음, 새로운 텍스트가 긍정적 표현인지 부정적 표현인지를 판단할 수도 있다. 혹은 주식시장에 긍정적이었던 뉴스기사와 부정적이었던 뉴스기사를 training text로 훈련시킨 다음, 새로운 뉴스가 떴을 때 증시에 긍정적인지 부정적인지 자동으로 판단하는 것도 가능하다.

이번 리포트에서는 긍정적인 영화리뷰 글과 부정적인 영화리뷰 글을 훈련시킨 다음, 새로운 영화리뷰 글이 들어왔을 때 긍정적인지 부정적인지를 자동으로 판단하는 모델을 샘플로 제시하고자 한다.[1]

나이브 베이즈 BoW 모델에서는 testing text에 들어있는 단어의 종류와 출현횟수에 의해서 클래스 분류의 판단이 이뤄진다. 대략 아래와 같은 방식이다.

$P(C_k|x_1, \dots, x_n)$의 예제: P(해당문장이 긍정적인 Class에 속함|"bad"라는 단어, … , "worse"라는 단어)

나이브 베이즈 BoW 모델에서는 $P(x_i|C_k)$의 확률 계산방식이 W/L모델의 경우와 다르다. 새로운 방식은 아래의 수식과 같다. BoW 모델에서는, 특정 문구가 해당 클래스의 전체 텍스트 중에서 몇 번의 빈도로 나왔는지의 값을 확률값으로 사용한다.

$$P(C_k|x_1, \dots, x_n) \propto P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$ (여기서 x_i는 각각의 단어를 뜻함)

$$P(x_i|C_k) = \frac{\text{해당 단어 어휘의 } C_k \text{ Training 문구전체 내에서의 출현횟수}}{C_k \text{ Training 문구전체의 총단어수}}$$

[1] 여기서의 Movie Review Data는 http://www.cs.cornell.edu/People/pabo/movie-review-data/에서 가져왔음

예를 들어, Testing text에 "bad"란 단어가 들어가 있다고 하자. 해당 속성에 의해서 Testing text가 긍정적인 감정 클래스에 속할지, 부정적인 감정 클래스에 속할지 판단하기 위해서 계산이 필요한 확률값은 $P("bad"|긍정\ Class)$와 $P("bad"|부정\ Class)$이다.

긍정적인 멘트들을 모은 긍정 Class Training 문구 내에서는 "bad"라는 단어가 아마도 적게 나올 것이다. 긍정 Class 텍스트집합의 전체 단어 개수가 100,000개고 이 중에 "bad"라는 단어가 2번만 나왔으면 $P("bad"|긍정\ Class) = 2/100{,}000 = 0.00002$가 된다. 부정 Class 텍스트집합에는 "bad"가 좀더 많이 나올 가능성이 있다. 부정 Class 텍스트집합의 전체 단어 개수가 100,000개고 이 중에 "bad"라는 단어가 130번 나왔다면 $P("bad"|부정\ Class) = 130/100{,}000 = 0.0013$로 계산된다. Testing text에 있는 각각의 단어들에 대해 위의 계산을 모두 해주고, 이 확률들을 모두 곱해주면, testing text가 어떤 Class에 해당하는 지를 판단할 수 있게 된다.

해당 확률 계산시에는 한가지 주의할 점이 있다. 만약 testing text 안에 "regrettable"이란 단어가 들어있는데, 부정 training text 전체문구 안에는 "regrettable"이란 단어가 한번도 없었다고 가정해보자 (충분히 가능성이 있는 일이다). 이때 해당 단어가 training text 문구 내에 없으므로, 이 때의 $P(x_i|C_k)$는 0으로 계산되고, 0이 한번이라도 들어가면 이를 포함하는 모든 곱셈은 항상 0이 된다. 몇몇 희귀한 단어로 인해 확률 계산이 0이 되는 것을 방지하기 위해, 처음 보는 단어라도 적당한 확률처리를 해주는 smoothing 과정이 필요하다. 이를 Laplace smoothing이라고 한다. Laplace smoothing을 포함한 확률 계산법은 다음과 같다.

$$P(x_i|C_k) = \frac{\text{해당 단어 어휘의 } C_k \text{ Training 문구전체 내에서 출현횟수} + \text{alpha}(0.2)}{C_k \text{ Training 문구전체의 총단어수} + \text{ 단어 어휘의 전체 개수} \times \text{alpha}(0.2)}$$

(여기서 alpha는 적당한 임의의 값이 가능함)

위의 수식 변화를 통해, 전혀 새로운 단어가 들어가더라도 매우 낮은 확률로 계산할 뿐, 0으로 계산되는 현상은 없어지게 된다.

이상으로, 나이브 베이즈 BoW 모델을 위한 준비운동은 끝났다.

BoW (Bag of Words) 모델 코드 및 설명

기계학습과 같은 알고리즘은 적정한 프로그래밍 언어를 기반으로 사용할 때 훨씬 효과적이다. 최근 기계학습에서 많이 사용하는 Language로는 Python과 R이 있다. 여기서는 Python 코드로 나이브 베이즈 BoW 모델을 소개하겠다. (여기 사용된 코드는 http://www.elice.io에서 많이 인용하였음. Python 언어는 무료로 사용할 수 있으며, 사용법을 쉽게 익힐 수 있음. 사용법 교육을 위해서는 http://wikidocs.net/book/1 등을 활용할 수 있음)

아래에는 Python(2.7 버전)으로 코딩된, 나이브 베이즈 Bow 모델의 전체 코드를 제시하였다.

프로그램 전체코드

```
# -*- coding: utf-8 -*-
#Python 2.7
import re
import math

def main():

    training1_text = file_load("D:/positive.txt")
    training2_text = file_load("D:/negative.txt")
    testing_text = file_load("D:/testing.txt")
    alpha = 0.2 # Laplace Smoothing용 계수
    prob1 = 0.5 # 영화평이 positive일지에 대한 사전적인 예상. 중립인 0.5
    prob2 = 0.5 # 영화평이 negative일지에 대한 사전적인 예상. 중립인 0.5

    # naive bayesian classifier 실행
    relative_prob_pair = naive_bayes_run(training1_text, training2_text, testing_text, alpha, prob1, prob2)

    # 결과 출력
    print("[testing_text]\n%s" % testing_text)
    print("\n[BOW data of testing_text]\n%s" % str(create_BOW(testing_text)))
    print("\n[probability of positive/negative]")
    print("pos:%s   nega:%s" % (relative_prob_pair[0],relative_prob_pair[1]))
    print("\n[Result]")
    if relative_prob_pair[0]>=relative_prob_pair[1]: print("Positive")
    else: print("Negative")


def file_load(file):
    f = open(file,'r')
    txtfile = f.read()
    f.close()

    return txtfile


def naive_bayes_run(training1_sentence, training2_sentence, testing_sentence, alpha, prob1, prob2):
    # testing_sentence를 positive문구들과 비교
    classify1 = math.log(prob1) + calculate_doc_prob(create_BOW(training1_sentence), create_BOW(testing_sentence),
alpha)
    # testing_sentence를 negative문구들과 비교
    classify2 = math.log(prob2) + calculate_doc_prob(create_BOW(training2_sentence), create_BOW(testing_sentence),
alpha)

    # positive확률과 negative확률을 상대확률 형태로 return
    return prob_adjustment(classify1, classify2)


# testing문구를 training문구와 비교하여, 유사성을 log(확률값)으로 표시하는 함수
```

```
def calculate_doc_prob(training_model, testing_model, alpha):
    logprob = 0

    # training문구의 단어출현수 합계
    num_tokens_training = sum(training_model[1])
    # training문구의 단어종류 개수
    num_words_training = len(training_model[0])

    # testing문구의 단어사전별로 진행
    for word in testing_model[0]:
        # 해당 단어의 testing문구 내 출현횟수
        word_freq = testing_model[1][testing_model[0][word]]
        # 해당 단어의 training문구 내 출현횟수
        word_freq_in_training = 0
        if word in training_model[0]:
            word_freq_in_training = training_model[1][training_model[0][word]]

        # 유사성 확률은 "해당 단어 어휘의 training문구 내 출현횟수/training문구의 전체 단어사용수 합계"로 계산함
        # 새로운 단어일 때 확률이 0이 되는 것을 막기 위해 Laplace Smoothing(alpha)을 사용함.
        # 모든 출현단어별로 확률값의 곱을 실행함. 확률값을 log로 변환하여 계산
        for i in range(0, word_freq):
            logprob = logprob + math.log(word_freq_in_training + alpha) - math.log(num_tokens_training +
num_words_training * alpha)

    return logprob


# BOW리스트자료를 생성함
# BOW리스트자료는 단어사전(bow_dict)과, 단어출현횟수정보(bow)로 이루어짐
# bow_dict은 각 단어별로 index를 지정한 Dictionary자료. bow는 단어index별로 출현횟수를 기록하는 리스트자료임
def create_BOW(sentence):
    bow_dict = {}
    bow = []

    sentence = sentence.lower()
    # 알파벳이외의 문자는 모두 빈칸으로 변환하는 정규식 코드
    sentence = re.sub(r'[^a-z]+', ' ', sentence)
    words = sentence.split(' ')
    for token in words:
        if len(token) < 1: continue

        # 신규단어가 나타나면, bow_dict에 추가하고, bow에도 0으로 추가
        if token not in bow_dict:
            new_idx = len(bow)
            bow.append(0)
            bow_dict[token] = new_idx

        # bow리스트내의 해당단어 count를 1 올려줌
        bow[bow_dict[token]] += 1

    return bow_dict, bow


# 2개의 확률수치에 대해, 전체합이 1이 되도록 상대확률로 표현하는 식
def prob_adjustment(prob1, prob2):
    pro1 = math.exp(prob1) / (math.exp(prob1)+math.exp(prob2))
    pro2 = math.exp(prob2) / (math.exp(prob1)+math.exp(prob2))

    return (pro1, pro2)


if __name__ == "__main__":
    main()
```

위의 코드에서 사용하는 외부 Input data는 positive.txt, negative.txt, testing.txt 3개 파일이다.

positive.txt는 긍정적인 평가가 들어있는 1,000개의 장문의 영화리뷰글을 모아놓은, 160만개의 단어로 이루어진 8메가 용량의 text file이다[2](코넬대학교 데이터를 정리해 놓았음. 영어 자료). negative.txt는 부정적인 평가가 들어있는 1,000개의 장문의 영화리뷰글을 모아놓은 140만개 단어/ 8메가 용량의 text file이다[3]. testing.txt는 **"This movie was really really bad... haven't seen worse in my life"**의 문구가 들어있는 간단한 testing문구 파일이다.

프로그램 실행결과

```
[testing_text]
This movie was really really bad... haven't seen worse in my life

[BOW data of testing_text]
({'life': 11, 'worse': 8, 'this': 0, 'movie': 1, 'bad': 4, 't': 6, 'my': 10, 'in': 9, 'seen': 7, 'was': 2,
'haven': 5, 'really': 3}, [1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1])

[probability of positive/negative]
pos:0.0566200061943   nega:0.943379993806

[Result]
Negative
```

3개 파일을 지정된 폴더(D:₩)에 넣고, 프로그램 코드를 실행하면 위와 같은 실행 결과가 나온다. "This movie was ..."가 들어가 있는 testing text를 긍정 및 부정 training text 집합과 비교해본 결과, 긍정적인 문구와는 5.7%의 상대확률 유사성, 부정적인 문구와는 94.3%의 상대 확률 유사성을 보였다. 결론적으로는 해당 문구는 부정적인 문장에 가깝다고 컴퓨터가 판단결과를 내렸다.

전체 프로그래밍 코드에 대해, 파트를 나눠서 간략히 설명하면 다음과 같다.

[2] http://goo.gl/aUymdE 에 positive.txt 파일을 업로드함. 다운로드 가능
[3] http://goo.gl/i03TSt 에 negative.txt 파일을 업로드함. 다운로드 가능

코드 #1

```
# -*- coding: utf-8 -*-
#Python 2.7
import re
import math

def main():

    training1_text = file_load("D:/positive.txt")
    training2_text = file_load("D:/negative.txt")
    testing_text = file_load("D:/testing.txt")

    alpha = 0.2 # Laplace Smoothing용 계수
    prob1 = 0.5 # 영화평이 positive일지에 대한 사전적인 예상. 중립인 0.5
    prob2 = 0.5 # 영화평이 negative일지에 대한 사전적인 예상. 중립인 0.5

    # naive bayesian classifier 실행
    relative_prob_pair = naive_bayes_run(training1_text, training2_text, testing_text, alpha, prob1, prob2)

    # 결과 출력
    print("[testing_text]\n%s" % testing_text)
    print("\n[BOW data of testing_text]\n%s" % str(create_BOW(testing_text)))
    print("\n[probability of positive/negative]")
    print("pos:%s   nega:%s" % (relative_prob_pair[0],relative_prob_pair[1]))
    print("\n[Result]")
    if relative_prob_pair[0]>=relative_prob_pair[1]: print("Positive")
    else: print("Negative")
```

Main 구문이다. 첫부분은 정규식 관련 패키지(re)과 수학관련 패키지(math)를 참조한다는 뜻이다(import 구문). 다음으로는 프로그램에 사용되는 text file를 불러와, 긍정적인 텍스트집합 전체 문자열을 training1_text 변수에 넣고, 부정적인 텍스트집합 전체 문자열을 training2_text 변수에 넣는다. 그리고 몇 개의 계수를 가정하고 나이브 베이즈 분류기를 실행한 다음에, 결과를 출력하는 부분으로 이어진다. 구체적인 알고리즘 계산은 뒤에 이어지는 하위 모듈에서 이루어진다.

코드 #2

```
def file_load(file):
    f = open(file,'r')
    txtfile = f.read()
    f.close()

    return txtfile
```

file_load 모듈은 지정된 파일명을 가지고 해당 텍스트파일을 읽어서, 그 내용 전체를 문자열로 리턴하는 역할을 한다.

코드 #3

```
def naive_bayes_run(training1_sentence, training2_sentence, testing_sentence, alpha, prob1, prob2):
    # testing_sentence를 positive문구들과 비교
    classify1 = math.log(prob1) + calculate_doc_prob(create_BOW(training1_sentence), create_BOW(testing_sentence),
alpha)
    # testing_sentence를 negative문구들과 비교
    classify2 = math.log(prob2) + calculate_doc_prob(create_BOW(training2_sentence), create_BOW(testing_sentence),
alpha)

    # positive확률과 negative확률을 상대확률 형태로 return
    return prob_adjustment(classify1, classify2)
```

나이브 베이즈 분류기의 기본 시작 모듈이다. training1_sentence(긍정적인 영화리뷰 문자열, 8메가짜리), training2_sentence(부정적인 영화리뷰 문자열), testing_sentence(테스트할 문자열)을 input으로 받는다. testing_sentence가 긍정 Class일 확률의 log값이, calculate_doc_prob 모듈을 통해 계산되여 classify1에 저장된다 (연속된 확률의 곱 수치가 소수점 아래자리로 크게 내려가기 때문에, log값으로 변환한 수치를 사용한다). testing_sentence가 부정 Class일 확률의 log값이 classify2에 저장된다. classify1, classify2를 Input으로 하는 prob_adjustment 모듈을 호출하고 그 결과를 리턴한다.

코드 #4

```
# testing문구를 training문구와 비교하여, 유사성을 log(확률값)으로 표시하는 함수
def calculate_doc_prob(training_model, testing_model, alpha):
    logprob = 0

    # training문구의 단어출현수 합계
    num_tokens_training = sum(training_model[1])
    # training문구의 단어종류 개수
    num_words_training = len(training_model[0])

    # testing문구의 단어사전별로 진행
    for word in testing_model[0]:
        # 해당 단어의 testing문구 내 출현횟수
        word_freq = testing_model[1][testing_model[0][word]]
        # 해당 단어의 training문구 내 출현횟수
        word_freq_in_training = 0
        if word in training_model[0]:
            word_freq_in_training = training_model[1][training_model[0][word]]

        # 유사성 확률은 "해당 단어 어휘의 training문구 내 출현횟수/training문구의 전체 단어사용수 합계"로 계산함
        # 새로운 단어일 때 확률이 0이 되는 것을 막기 위해 Laplace Smoothing(alpha)을 사용함.
        # 모든 출현단어별로 확률값의 곱을 실행함. 확률값을 log로 변환하여 계산
        for i in range(0, word_freq):
            logprob = logprob + math.log(word_freq_in_training + alpha) - math.log(num_tokens_training +
num_words_training * alpha)

    return logprob
```

나이브 베이즈 분류기의 핵심적인 부분이다. 여기서의 Input변수인 training_model, testing_model은 윗단계 모듈인 naive_bayes_run 안에서 create_BOW (training1_sentence), create_BOW (testing_sentence)로 사용된 것이다. create_BOW 모듈을 통해서 BoW자료 형태으로 변환된 결과를 Input 변수로 받고 있는 셈이다. Bow자료형 설명 및 변환 과정은 다음의 create_BOW 모듈에서 자세히 볼 수 있다.

BoW자료 형태인 testing_model에서는 testing_model[0]이 단어 Dictionary자료, testing_model[1]이 단어 출현횟수 기록 리스트에 해당한다. testing 문구의 각 단어별로 진행을 하여, 각각의 $P(x_i|C_k)$(11페이지 참고)를 계산한다. 계산된 $P(x_i|C_k)$를 log값으로 변환한 다음 모든 단어에 대해서 합산하여 최종적으로 해당 Class에 속할 확률값(log확률값)을 리턴한다.

코드 #5

```
# BOW리스트자료를 생성함
# BOW리스트자료는 단어사전(bow_dict)과, 단어출현횟수정보(bow)로 이루어짐
# bow_dict은 각 단어별로 index를 지정한 Dictionary자료. bow는 단어index별로 출현횟수를 기록하는 리스트자료임
def create_BOW(sentence):
    bow_dict = {}
    bow = []

    sentence = sentence.lower()
    # 알파벳이외의 문자는 모두 빈칸으로 변환하는 정규식 코드
    sentence = re.sub(r'[^a-z]+', ' ', sentence)
    words = sentence.split(' ')
    for token in words:
        if len(token) < 1: continue

        # 신규단어가 나타나면, bow_dict에 추가하고, bow에도 0으로 추가
        if token not in bow_dict:
            new_idx = len(bow)
            bow.append(0)
            bow_dict[token] = new_idx

        # bow리스트내의 해당단어 count를 1 올려줌
        bow[bow_dict[token]] += 1

    return bow_dict, bow
```

문자열 정보를 받아 BoW 자료형태를 리턴하는 모듈이다. Input 문자열을 먼저 소문자로 변환한다. 그 다음에 re.sub 모듈을 사용해서 비알파벳문자를 모두 스페이스로 변환한다. 이는 정규표현식(Regular Expression) 기능을 통해서 한 줄로 구현되어 있다. 최종 정리된 문자열을 가지고, 첫째, 각 단어에 대해서 단어를 Key로 가지고, 단어의 위치(단어 index)를 Value로 가지는 Dictionary 자료를 만든다 (Dictionary는 Key:Value 쌍들을 기록하는 Python의 자료형임). 둘째, 해당 단어가 반복해서 사용된 횟수를 단어 index에 기록하는 리스트 자료를 만든다. 마지막으로, 단어 위치 Dictionary(bow_dict)와 단어 출현 횟수 리스트(bow)를 최종리턴한다.

예제로 사용된 Testing text를 BoW자료 형태로 변화하면 다음과 같이 나온다.

```
원 텍스트 = “This movie was really really bad... haven't seen worse in my life”
BoW 자료형태 =
({'life': 11, 'worse': 8, 'this': 0, 'movie': 1, 'bad': 4, 't': 6, 'my': 10, 'in':
9, 'seen': 7, 'was': 2, 'haven': 5, 'really': 3}, [1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
1])
```

예제를 보면, 원 텍스트에서 really라는 단어가 두 번 사용되었다. BoW 자료형태의 첫번째 부분인 Dictionary를 보면 ‘really’: 3 즉, really의 단어위치가 3으로 나온다. BoW 자료행태의 두번째 부분인 출현 횟수 리스트를 보면, [1, 1, 1, 2, ...]로 나와 리스트의 3번(네번째) 인덱스가 2로 기록된 것을 알 수 있다. (프로그래밍에서는 항상 인덱스가 0에서부터 시작한다)

코드 #6

```
# 2개의 확률수치에 대해, 전체합이 1이 되도록 상대확률로 표현하는 식
def prob_adjustment(prob1, prob2):
    pro1 = math.exp(prob1) / (math.exp(prob1)+math.exp(prob2))
    pro2 = math.exp(prob2) / (math.exp(prob1)+math.exp(prob2))

    return (pro1, pro2)
```

긍정적인 Class일 확률값의 log수치, 부정적인 Class일 확률값의 log수치를 받아서, 합계가 100%인 상대적인 확률값으로 바꿔주는 모듈이다. 우선 exp함수를 써서 log값을 원값으로 변환하고, 그 다음에 합계가 100%이 되도록 비율을 계산하였다.

코드 #7

```
if __name__ == "__main__":
    main()
```

메인 모듈 실행을 위한 기계적인 문구다.

BoW (Bag of Words) 모델 활용

이상으로 나이브 베이즈 BoW모델 코드의 설명이 끝났다. Python을 통해 주석까지 포함하여 '단 110줄의 코드' 만으로 텍스트 분석 알고리즘을 실행할 수 있게 되었다. 해당 프로그램은 적절한 training text만 주어진다면, 다방면의 텍스트 자동 분석이 가능하다. Training text로 증시에 긍정적인 뉴스기사, 부정적인 뉴스기사들을 학습시킨 다음, 실시간으로 올라오는 뉴스를 긍정적인지 부정적인지 판단하는 모델로 사용할 수도 있다. 최근에는 한글 자연어처리 패키지 등도 많이 개발되고 있으므로, 한글 텍스트 처리도 어느 정도 가능할 것으로 생각된다. 또한 DART(전자공시시스템)에 올라온 공시 정보를 긍정적인 정보, 부정적인 정보로 나눠서 학습한 다음, DART의 신규 공시에 대해 자동 분석하는 방법도 가능하다. DART에서도 실시간 정보조회를 위한 API를 공개하고 있어, 프로세스 전과정을 자동화하는 것도 불가능한 일이 아니다.

Contents

V. 결론

이상으로, 기계학습 알고리즘의 중의 하나인 Naive Bayes Classifier에 대해 소개하고 주식투자 전략에 접목하는 방법에 대해 그 예를 제시했다.

선진국의 비즈니스 환경에서는 빅데이터, 기계학습이 이미 커다란 추세로 자리잡고 있다. 선진국 금융투자업계에서도 기계학습 알고리즘을 사용하는 사례가 점차 늘고 있다. 로보어드바이저 서비스, 다양한 퀀트 펀드 들이 기계학습 알고리즘 채택을 늘리고 있다. 국내 금융투자업계에서도 기계학습을 사용하는 새로운 투자 방법에 대해 점차 대비를 할 필요가 있다고 판단되는 바이다.

Naive Bayes Classifier의 2가지 모델인 W/L 모델과 Bow 모델 중에서, 현재 주식 투자에 바로 활용할 수 있는 것은 W/L 모델이다. W/L 모델과 최근 2년간의 주가 흐름의 구조를 반영하여, 단기적인 주가상승 예상종목과 주가하락 예상종목을 구별해보면 아래 표와 같다.

아래 표에서의 Winner가 퀀트로 뽑은 단기적인 매수추천종목, Loser가 매도추천종목에 해당한다.

W/L모델: 시가총액 상위 20 종목의 투자 판단

종목명	Book to Price (%)	매출액 증가율 (%)	TP괴리율 (%)	주가변동성 (%)	ROE (%)	최종 클래스
Winner						
한국전력	223.4	0.6	34.9	25.3	9.1	Winner
삼성물산	66.6	70.4	49.7	52.0	2.8	Winner
아모레퍼시픽	13.7	20.9	21.1	36.9	20.9	Winner
SK하이닉스	109.6	-3.1	44.0	31.5	14.2	Winner
SK	68.6	86.1	34.7	36.1	13.3	Winner
SK텔레콤	109.0	2.4	57.6	25.4	11.6	Winner
POSCO	291.7	0.5	46.0	26.7	3.4	Winner
KB금융	231.9	6.9	52.5	22.2	5.7	Winner
NAVER	18.3	14.9	33.4	32.6	25.8	Winner
신한지주	166.1	5.1	43.5	23.3	7.4	Winner
Loser						
삼성전자	94.0	1.6	31.2	24.2	11.4	Loser
현대차	165.8	3.5	33.3	32.9	10.6	Loser
현대모비스	121.3	5.8	25.8	25.8	12.4	Loser
기아차	127.8	5.9	25.4	26.2	12.6	Loser
LG화학	58.7	6.9	15.8	34.7	11.5	Loser
삼성생명	120.2	2.8	28.3	24.1	5.6	Loser
삼성에스디에스	25.1	11.2	27.6	55.9	10.6	Loser
LG생활건강	13.9	12.7	13.3	38.1	25.0	Loser
KT&G	48.6	-0.2	28.9	24.6	14.7	Loser
삼성화재	76.8	2.9	19.0	24.3	9.5	Loser

참고: 여기서 Winner, Loser 기준은 향후 3개월간 주가수익률이 KOSPI를 아웃퍼폼 하는지를 기준으로 삼음
계량적 분석에 의한 판단이며, 당사 기업분석 투자의견과 다를 수 있음

자료: 삼성증권

Compliance notice

- 본 보고서는 철저히 계량적 분석에 근거한 의견을 제시합니다. 따라서 당사의 대표 투자의견과 다를 수 있습니다.
- 본 조사분석자료는 기관투자가 등 제 3자에게 사전 제공된 사실이 없습니다.
- 본 조사분석자료에는 외부의 부당한 압력이나 간섭없이 애널리스트의 의견이 정확하게 반영되었음을 확인합니다.
- 본 조사분석자료는 당사의 저작물로서 모든 저작권은 당사에게 있습니다.
- 본 조사분석자료는 당사의 동의없이 어떠한 경우에도 어떠한 형태로든 복제, 배포, 전송, 변형, 대여할 수 없습니다.
- 본 조사분석자료에 수록된 내용은 당사 리서치센터가 신뢰할 만한 자료 및 정보로부터 얻어진 것이나, 당사는 그 정확성이나 완전성을 보장할 수 없습니다. 따라서 어떠한 경우에도 본 자료는 고객의 주식투자의 결과에 대한 법적 책임소재에 대한 증빙자료로 사용될 수 없습니다.

삼성증권 SAMSUNG

삼성증권주식회사
100-742 서울특별시 중구 세종대로 67 삼성본관빌딩 12층 리서치센터
02 2020 8000

지점 대표번호
1588 2323 / 1544 1544

고객 불편사항 접수
080 911 0900

samsung POP.com