thwk2019. 12.9 (월)

안녕하세요? 금년의 마지막 달 12월입니다. 저번 주에는 이곳에 눈이 펑펑 내리기도 했는데 겨울을 실감하게 해준 것 같습니다. 그리고 기온이 영하 9도로 꽁꽁 얼어붙을 것 같았었죠. 정말로 본격적인 추위가 온 것 같습니다. 어떻게 지내시는지요? 추위에 항상 건강하시길 빕니다.

며칠 전 도장생기3팀 김희수 책임님으로부터 연락을 받았습니다. 데이터 분석그룹 모임이 오늘부터 시작되었는데 멀리 광주에서도 참석을 하고자 한다는 내용이었습니다. 그래서 첫 모임은 서로 얼굴 보면서 앞으로 어떻게 운영할지에 대해 편안하게 이야기를 나누는 시간이라서 그 다음 모임부터 본격적 모임이 시작될 것이라고 답변 드렸습니다. 멀리서도 모임에 참여할 수 있게 인강(인터넷강의)이라도 만들어서 보내야 하는 것 아니냐고 우스개 소리로 말씀하셨는데.. 사실 저는 그 정도로 관심을 가지고 계신지도 모르고 있었던 터라 약간 놀랐습니다.^^ 필요하면 광주에서 출장을 가든지, 아니면 영상회의로라도 원격모임을 갖길 원한다고 거듭 강조하셔서 어찌할지 사실 많이 고민이 되었습니다. 일단 데이터 분석 소그룹 모임이 생기1실을 대상으로 하기 때문에, 그렇게 한정지어서 운영은 하도록 하겠습니다. 하지만 모임 일정이라든지, 모임에서 다뤄진 내용은 제가 계속적으로 정리해서 공유하도록 하겠습니다. 그리고 모임에서 다뤄지거나 따로 공부하실 수 있는 자료는 보내드리겠습니다. 원격으로 영상회의를 갖으면서 코딩을 하며 설명하는 것은 생각해 보겠습니다. 한다고 말씀드리는 것은 아니고요^^;; 생각은 해보겠습니다. 무엇보다 큰 관심을 가져주신 것에 대해 큰 감사의 말씀을 드립니다.

데이터 분석 소그룹 모임은 저번 주 모집을 마지막으로 최종 총 22명이 모이게 되었습니다. 차체부품생기팀 4명, 프레스생기팀 3명, 차체생기1팀 2명, 도장생기1팀 7명, 차량생기1팀 6명입니다. 저와 백인택 책임까지 합하면 총 24명이 됩니다. 모임은 매주 금요일 오후 13:30부터 2시간 가량 갖게 됩니다. 장소는 당분간은 소하리 생기센터 5층 겨울왕국 회의실이고 실습이 이뤄지게 되면 소하교육센터로 옮길 예정입니다.

생기센터 5층 LAY-OUT

①

②

③

④

⑤

⑦

⑥

⑧

차량생기1팀

차량생기1팀

도장생기1팀

입출구

[범례]

1. 토이스토리
2. 주토피아
3. 코난
4. 피카츄
5. 휴게실
6. 미니언즈
7. 겨울왕국
8. 협업라운지

회의실 이름은 제 임의대로 애니메이션 영화 이름으로 정해 보았습니다.^^ 회의실 위치를 말하라고 할 때, 정확한 이름을 대는 것이 없이 그냥 회의실이니까요. 회의실1,2,3,4... 뭐 이런 식으로도 정해도 되지만 회의실에 고유대명사가 있으면 바로 연상이 되니까 편할 것 같아서 만들어 봤습니다. 차량생기팀 분들께서 다른 이름이 있으면 바꿔서 알려 주시면 그걸로 하겠습니다.^^ 뭐 많이 사용하는 것이 아니라서 대충 해봤습니다. 소모임은 항상 겨울왕국에서 이뤄질 예정이니 겨울왕국으로 오세요.^^

그리고 22명을 A~E까지 5개조로 나눴는데, 팀간 울타리를 넘어서 서로 가르쳐 주고 함께 하고자 하는 이유입니다. 화성에서 오는 차체부품생기팀을 한 조로 묶었고, 생기센터 4층에 있는 프레스생기팀과 차체생기1팀 인원을 한 조로 묶었고, 5층은 인원은 많아서 3개조로 나눠서 섞었습니다. 차체부품생기팀만 제외하고 모든 팀이 섞였다고 보시면 되겠습니다. 조로 편성한 것은 어찌보면 같이 함께 하기 위한 약간 형식적인 것이라고 보면 되고요, 무엇보다 중요한 것은 서로 울타리 없이 아주 친하게 지내는 것이 가장 중요하다 생각합니다. 필요시 MT를 가거나 양지 연수원을 날을 잡아서 밤새 공부하거나 또는 밤새 놀 수도 있습니다.^^ 그만큼 편하게 생각해 주셨으면 합니다.

사실 함께가 안 된다면 모든 것이 허사가 될 수도 있다고 생각합니다. 어디를 가든지 이 ‘함께’가 가장 어려운 것이아닌가 싶습니다. 데이터 분석 능력이 향상되어 어떤 과제를 수행하고 적용하는 것이 어찌보면 누구나 상상하고 바라는 우리 모임의 예상모습이겠지만, 저는 약간 생각이 다릅니다.

나름대로의 단기적, 단회적 과제수행은 팀이나 실에는 어떤 가시적인 영향을 줄지 모르겠지만, 데이터 분석문화의 회사차원의 장기적인 저변확대에는 크게 영향을 주지 못한다는 것이 제 생각입니다. 소모임으로 시작했지만 소모임이 또 다른 소모임을 만들고 그 소모임이 또 다른 소모임을 만드는 적어도 3세대에 걸친 소모임의 확대가 이뤄질 정도의 ‘함께’라는 샘물이 소모임으로부터 계속 샘솟지 못하게 된다면 소모임은 내부적으로, 또는 외부적인 누군가의 힘에 의해 곧 문을 닫게 될 것이 분명하기 때문입니다. 그래서 사실 저는 데이터 분석 능력향상에도 스스로도 많은 개인적인 노력을 하겠지만, 회사에서 그 누구도 해보지 않았던 ‘함께’의 파워가 어떤 것인지 맛보고 싶기도 합니다. ‘함께’는 누군가의 프로그램적으로, 어떤 인위적으로, 지시적으로, 계획적으로, 한 개인의 희생으로도 절대로 불가능합니다. 어떻게 하면 3세대에 걸친 소모임을 낳는지는 계속 연구해야 하겠지만 제가 알고 있는 한 가지 확실한 것은 있습니다. 그건 아래 두 가지 단어에 있습니다. 이것은 방법의 일부이지 전부는 아닙니다. 여러분들께서 모두 아시는 단어입니다.

공유, 가족

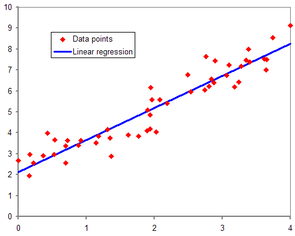
저는 항상 소모임 운영에 대해 이 두 단어를 품고자 합니다. 물론 그 동안 제가 해왔던 것이기도 합니다. 이 시대는 공유와 연결에 목말라 가고 있고, 또한 가족이라는 개념이 점점 사라지고 있습니다. 그 동안 ‘함께’라는 샘물이 샘솟지 못한 이유도 많은 이유 중에서 저 두 단어에 원인이 있었음을 저는 어느덧 깨닫게 되었습니다. 회사라는 이해집단이 항상 무슨 일을 할 때, ‘함께’에 목말른 이유도 회사내 오픈된 ‘공유’문화에 대한 시대적 변화, 오픈된 ‘가족’적 조직에 대한 시대적 변화가 계속 요구되기 때문입니다. 공유되지 않는 곳에는 ‘함께’라는 샘물은 메마르고 사라지게 되고, 가족적이지 않은 곳에는 무서운 개인주의가 곰팡이 피듯 번지게 됩니다. 요즘 회사내 조직간 이기주의를 말하는 사일로(silo)현상이라는 용어가 대두되는 이유도 바로 그 일환이라고 할 수 있습니다. 회사는 이것을 막기 위해 화합과 경쟁이라는 이름으로 막대한 돈을 들여서 여러가지 프로그램을 계획하고 교육을 하고 사람을 고용하게 되죠. 그 동안 해오지 않았던 ‘사람’을 연구하지 않는다면 인공지능시대의 그 인공지능은 냉정하고도 무서운 칼로 우리를 오히려 위협하게 될 것입니다. 인공지능은 절대로 ‘함께’라는 샘을 만들 수 없기 때문입니다.

요컨대, 훗날 소모임의 데이터 분석능력이 향상되어 과제를 수행/적용하게 되는 모두가 바라는 그런 모습은 ‘함께’라는 나무에 열리게 되는 한 개의 열매일 뿐, 그 열매가 전부일 수는 없다는 것입니다. 앞으로 소모임이 키우게 될 ‘함께’라는 나무의 열매만 보시는 분들은 앞으로 소모임을 망칠 수 있는 잠재적인 능력을 가지신 분들이 될 것입니다. 소모임이 ‘함께’라는 나무를 잘 키울 수 있게 거름을 주시고, 추위를 막아 주시고, 필요시 병충해를 방지하기 위해 소독을 해 주실 수 있는 분들이 많아지는 것이 저의 바램입니다. 그리고 그 나무가 또 새로운 나무를 낳을 수 있도록 계절이 지나 씨가 뿌려질 수 있게 많은 노력과 열정이 있어야 할 것이고요, 그렇게 세대를 거듭하여 여유가 생긴다면 품종이 좋은 나무를 가져와 가지접붙이기 시도를 해서 열매의 당도와 품질을 높을 수도 있는 그런 큰 그림도 상상해 봅니다.

또 말이 길어졌습니다. ^^ 아무튼 지원 참여해 주신 22명의 각 팀원분들께 다시 한 번 감사드립니다. 저도 항상 배우는 자세를 잃지 않고, 어미새가 새끼새에게 먹이를 물어다 주는 마음으로 모르는 것 빼고 아는 한도 내로 최대한 지원을 하도록 하겠습니다.

그럼 오늘부터는 데이터 뽀개기 4편인데, 저번까지는 통계부문에서 골프스코어로 분산분석 약간 맛을 봤고요, 상관관계에 대해서 예시를 통해서 설명해 드렸는데, 오늘은 아주 중요한 회귀분석에 대해서 알려 드리고자 합니다. 좀 길 수도 있지만 예측분석부문에서 어떻게 보면 가장 큰 비중을 차지하고 있는 부분이 아닌가 생각합니다. 연속형 양적변수들의 상관관계를 모델링을 해서 그 모델을 가지고 앞으로 들어오는 새로운 값을 x값을 대입하여 y값을 예측하는 지도학습 머신러닝이라고 보시면 됩니다. 과거에 가지고 있던 x와 y값들을 가지고 직선 또는 곡선을 그려서 상관관계를 표현하고 그렇게 표현한 직선 또는 곡선 모델식을 가지고 미래의 데이터에 대해 예측하는 것인데, 기본적으로 어떤 식으로 모델링을 하는지 원리를 익힐 필요가 있겠습니다. 그리고 생성된 모델에 대해서 평가하는 법도 알아보겠습니다.

기본적인 선형회귀분석을 가지고 원리를 설명할 때, 항상 최소제곱법(Least Square Method)이 나옵니다. 대학교때, 실험을 하고 그 결과값을 엑셀을 이용해서 그래프 상에 점을 찍고 그 위에 가장 적합한 선을 그려서 선의 식을 구하는 것을 많이 했는데, 그 식이 바로 그 실험결과를 말하는 모델식이라고 보시면 됩니다.



빨간 점은 실험결과 데이터이고 파란선은 그 결과를 나타내주는 직선 모델식이 됩니다. 이 모델식이 있으면 x축 실험값에 대한 y축의 예측값을 예측할 수가 있죠. 물론 모델식이 납득할 수 있을만큼 적합해야 정확한 예측값을 얻을 수가 있겠죠. 그럼 모델식인 파란선을 어떤 원리로 만들어 내는가에 대해서 최소제곱법에 대해 알아보겠습니다.

실험 관찰치 가 만약 우리가 모델식으로 만들고자 하는 y = ax + b 직선 상에 있다고 가정하면, 식은 다음과 같습니다.

모델식은 위와 같이 되지만, 실제로는 실험관찰치 각 점에서 만큼 오차(잔차)가 있습니다. 오차만큼 떨어져 있는 것입니다. 각 점에서 가장 적게 떨어져 있게 모델선을 찾아내면 그 선이 실험 관찰치를 가장 잘 나타내 주는 식이 되겠습니다.

여기에서 를 전체로서 가장 작게 만들면 되는데 음수값이 나올 수도 있고 양수값이 나올 수도 있기 때문에 그 대신 을 합계한 값을 최소로 하면 되겠습니다.

D를 최소로 하는 a와 b를 구하려면, D를 a와 b로 편도함수, 즉 미분값이 0이 될 경우가 되겠습니다.

*→*

*→ →*

→

어디에서 많이 보던 식이 나왔습니다. 위 b값은 오차가 최소가 되게 하는 값이 되겠습니다. b값을 구하면 당연히 a값도 나오게 되죠. 예시로 한 번 설명드리겠습니다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *x* | *2* | *3* | *4* | *5* | *6* |
| *y* | *12* | *7* | *8* | *5* | ***3*** |

실험을 통해서 x와 y의 값이 위와 같이 얻게 되었다고 가정하겠습니다. b값을 구하기 위해서 계산에 필요한 값을 아래 표와 같이 정리했습니다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *x* | *y* |  |  |  |  |
| *2* | *12* | *-2* | *5* | *4* | ***-10*** |
| *3* | *7* | *-1* | *0* | *1* | ***0*** |
| *4* | *8* | *0* | *1* | *0* | ***0*** |
| *5* | *5* | *1* | *-2* | *1* | ***-2*** |
| *6* | *3* | *2* | *-4* | *4* | ***-8*** |
| 합계 | *20* | *35* | *0* | *0* | *10* | ***-20*** |

회귀직선 모델식은 로 구해졌습니다. 그런데 추정된 회귀직선은 모든 점을 통과하지는 않습니다. 거의 모든 점을 통과할 수준으로 구해졌다고 볼 수 있지 모든 점을 통과하지는 않는 것이죠. 회귀직선이 모델된 기준이 ‘오차제곱합의 최소’였기 때문입니다. 그러니까 오차제곱합이 최소가 되는 최적의 모델직선인 것입니다. 그래서 오차제곱합을 모델링 적합도(closeness of fit)의 척도라고 부릅니다. 나중에 머신러닝을 할 경우에 이 적합도에 대한 얘기를 많이 하는데, 과대적합(overfitting)과 과소적합(underfitting)이 있습니다. 나중에 직선이 아닌 다항회귀인 곡선으로 모델링을 할 경우, 아래와 같이 복잡도를 높여서 모델링을 하면 모든 점을 통과하는 선을 만들 수가 있습니다. 그러나 이렇게 하는 것은 모델의 일반화와 거리가 멀어 분석가들은 피합니다. 과대적합이 되면 테스트용 데이터를 모델에 적용했을 경우에 예측도가 떨어지는 단점이 있기 때문입니다. 모든 점을 통과했기 때문에 아주 잘 된 모델 곡선이라고 생각할 수 있지만 아래 그림에서 보듯이 4번째 점 다음에 임의의 5번째 점은 전혀 엉뚱한 위치에 찍힐 것입니다.

과소적합은 이와는 반대로 충분히 적합하게 모델곡선을 표현할 수 있는데 너무 단순하게 표현한 경우입니다.

점선으로 표현된 곡선으로 모델링을 할 수 있는데 너무 단순하게 직선으로 표현한 경우로 이것은 실험결과를 충분히 표현하지도 않고 예측도 불가능한 경우가 되겠습니다. 그래서 적합도의 변화를 잘 판단하면서 모델링을 해야 하는 것이죠. 제가 지금까지 예시로 든 것은 직선상관관계가 있는 직선회귀이기 때문에 곡선보다는 보다 쉬운 경우라 할 수 있습니다. 적합도를 표현하는 오차제곱합(=잔차제곱합)을 다음과 같이 나타낼 수 있습니다.

이것은 다음을 의미합니다.

[1]

x와 y가 전혀 관계가 없을 때는 계수 b는 0이 됩니다. x과 y가 관계가 있어야 기울기인 b도 변하게 되기 때문이죠. 하지만 상관관계가 없으니까 x가 아무리 변하여도 y는 증가하거나 감소하지 않게 됩니다. 다시 말하면 b=0일 때, 가 되고요, y값이 아무리 변하더라도 이것은 설명할 수 없는 변동이 되게 됩니다. 이것을 설명되지 않는 변동이라고 표현합니다. 바꿔서 말하면 반대로 는 설명되어야 할 변동이 되는 것이죠.

[2]

그럼 이번에는 만약 x와 y가 완전한 직선적 상관 관계가 있을 때, 회귀직선은 모든 점을 통과하게 됩니다. 그렇게 되면 오차제곱합은 당연히 이 되고, y의 모든 변동은 x에 의하여 설명될 수 있게 됩니다. 서로 완벽한 직선적 상관관계가 있기 때문이죠. 따라서 일 때, 가 되게 됩니다. 그렇다면 는 x에 의해 설명되는 y의 변동이라고 할 수 있습니다.

그러면 잔차제곱합을 이항해서 다시 표현해 보겠습니다.

이 식은 아래와 같이 다시 표현할 수 있습니다.

1 = 설명된 변동비 + 설명되지 않는 변동비

이와같이 x에 의해 설명되는 변동비를 결정계수(R squared, coefficient of determination)이라고 합니다. 이 결정계수를 R2으로 표현합니다. 결정계수값이 바로 종속변수 y에 대해서 설명변수 x의 설명력을 판단하는 기준이 됩니다. 바꿔 말하면 설명변수의 변동량으로 설명되는 종속변수의 변동량을 의미합니다. 선형회귀의 경우 결정계수는 또한 피어슨 상관계수의 제곱값과 동일합니다. 결국 피어슨 상관계수는 직선 상관관계를 설명하는 값이 되는 것이죠.

결정계수는 0~1의 사이의 값을 갖게 됩니다. 그럼 결정계수를 구해 보도록 하겠습니다. 결정계수를 구하기 위해서 필요한 값을 아래 표와 같이 정리해 보았습니다.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *x* | *y* |  |  |  |  |  |  |  |
| *2* | *12* | *-2* | *5* | *25* | *11* | *1* | *1* | *-10* |
| *3* | *7* | *-1* | *0* | *0* | *9* | *-2* | *4* | *0* |
| *4* | *8* | *0* | *1* | *1* | *7* | *1* | *1* | *0* |
| *5* | *5* | *1* | *-2* | *4* | *5* | *0* | *0* | *-2* |
| *6* | *3* | *2* | *-4* | *16* | *3* | *0* | *0* | *-8* |
| 합계 | *20* | *35* | *0* | *0* | *46* |  | *0* | *6* | *-20* |

(피어슨 상관계수)

0.93 상관계수값으로 볼 때 거의 완벽한 직선적 상관관계를 나타내고 결정계수가 0.87로 설명력도 매우 높음을 볼 수 있습니다. 이렇게 모델링을 한 직선을 가지고 테스트 x값을 대입하여 예측값 y를 구해내게 되는 것입니다. 이것이 예측인 것입니다. 그러나 모든 현상이 직선적으로 표현되면 좋겠지만 그렇지 않은 경우가 대부분이죠. 그래서 조금 더 복잡한 모델링이 필요한 것이고 이런 모델링이 가능하게 하는 알고리즘을 누군가가 편리하게 만들어 놓았기 때문에 이 알고리즘을 가져다가 최적합 모델링을 하는 것입니다.

그런데 제가 지금까지 말씀드리지 않았는데 선형회귀에서 확인해야 할 주의사항이 몇 가지 있습니다. 참고로 알아둘 필요가 있어서 말씀드립니다.

1. 설명변수인 x값이 측정오차로 인해 오염되지 않았어야 합니다.
2. 오차항의 평균값은 0이어야 합니다.
3. 수집된 x와 y값은 정규분포를 이뤄야 합니다.
4. 설명변수가 2개이상으로 많을 경우에 변수 서로간에 상관관계가 없어야 합니다.
5. 오차항은 모든 설명변수 x값에 대해서 동일한 분산을 가져야 합니다.

요약해서 말씀드리자면 실험에서 얻는 데이터가 다른 원인에 의해서 측정오차가 발생했을 경우에 데이터는 정규분포를 이루지 않을 수가 있습니다. 그리고 설명변수가 많을 경우, 설명변수(독립변수)간에 상관관계가 있으면 제대로 된 모델링이 될 수가 없게 됩니다. 이 정도만 확인해 주시면 될 것 같습니다. 나머지는 선형회귀에 해당하기 때문에 나중에 복합한 모델링을 할 경우에는 위 2가지 사항만을 확인해 주시면 되겠습니다. 정규성 확인은 이후에 다시 한 번 설명드리겠습니다. 그럼 단순선형회귀와 다항회귀를 한 번 비교해 보도록 하겠습니다. 다항회귀는 곡선으로 모델링을 하는 것인데 데이터가 있을 때 어떤 모델이 더 적합한지 확인해 보겠습니다. 참고로 다항회귀의 회귀 모델식은 아래와 같습니다.

이것을 일반적인 형태로 표현하면 다음과 같습니다.

다차원 다항식으로 두고 회귀분석을 수행하는 것을 다항회귀(polynomial regression)이라고 합니다. 다항회귀는 다중회귀 일종인데, 다중회귀(multiple regression)는 설명변수(독립변수)가 여러 개 있는 식에서 설명변수의 계수를 구하는 것을 말합니다. 로 치환을 하면 아래와 같이 다항회귀식 모델이 됩니다.

즉, 다항회귀 모델은 다중회귀모델로 계산될 수 있다는 의미가 됩니다. 그러면 주피터 노트북을 실행하시고 파이썬 코딩을 해 보겠습니다. 아래 코드를 셀에 입력하고 실행하겠습니다.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import pandas as pd

import numpy as np

from datetime import datetime

import datetime

from datetime import timedelta

import matplotlib as mpl

import matplotlib.pyplot as plt

from matplotlib.pyplot import cm

from matplotlib import pyplot

import matplotlib.font\_manager as fm

import seaborn as sns

from scipy import stats

from pandas.plotting import register\_matplotlib\_converters

register\_matplotlib\_converters()

# 노트북 안에 그래프를 그리기 위해

%matplotlib inline

# 그래프에서 격자로 숫자 범위가 눈에 잘 띄도록 ggplot 스타일을 사용

plt.style.use('ggplot')

# 그래프에서 마이너스 폰트 깨지는 문제에 대한 대처

mpl.rcParams['axes.unicode\_minus'] = False

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

from sklearn.metrics import r2\_score

from sklearn.preprocessing import PolynomialFeatures

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

맨 밑에 4줄만 빼고 나머지는 그 동안 분석할 때 기본적으로 불러온 모듈들 모음입니다. 맨 밑 4줄은 사이킷런(sklearn)이라는 모듈을 가져온 것인데 모듈에서 선형회귀(Linear Regression)함수를 가져왔고, 평균제곱오차(mean\_squared\_error,MSE), 결정계수(R2,r2\_score), 다항회귀(PolynomialFeatures)관련 함수를 가져왔습니다. 평균제곱오차라는 용어가 생소하겠지만 이미 위에서 우리가 언급한 내용입니다.

은 오차제곱합을 말하는데 이 값에 데이터수로 나누면 평균제곱오차가 되겠습니다. 아래와 같이 다시 정리할 수 있습니다.

MSE =

이 MSE가 최소가 되는 값이 가장 적합한 모델식이 된다는 의미가 됩니다. 그리고 MSE식을 손실함수(loss function)이라고 부릅니다. 회귀분석을 위해 데이터를 회귀분석 알고리즘에 넣고 모델을 만든 다음에 그 모델이 잘 만들어 졌는지 평가를 하는 기준으로 결정계수값이 1에 가까운지 확인하고, 손실함수값이 증가하는지 감소하는지 확인을 하는 작업을 해나가면서 모델을 최적화합니다. 그러면 임의로 데이터를 아래와 같이 만들어 봤습니다. 이번에는 코딩 내용을 한꺼번에 적어서 보내드립니다. 자세한 내용은 4편 두번째 시간에 설명을 드리겠습니다.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

X = np.array([258.0,270.0,294.0,320.0,342.0,

368.0,396.0,446.0,480.0,586.0])[:,np.newaxis]

y = np.array([236.4,234.4,252.8,298.6,314.2,

342.2,360.8,368.0,391.2,390.8])

lr = LinearRegression()

pr = LinearRegression()

# 다항 회귀를 위한 2차항 추가

quadratic = PolynomialFeatures(degree = 2)

X\_quad = quadratic.fit\_transform(X)

# 비교를 위해 단순회귀계산

lr.fit(X,y)

X\_fit = np.arange(250,600,10)[:,np.newaxis]

y\_lin\_fit = lr.predict(X\_fit)

# 다항회귀를 위해 변형된 모델에 다중회귀 모델 계산

pr.fit(X\_quad,y)

y\_quad\_fit = pr.predict(quadratic.fit\_transform(X\_fit))

# 단순회귀 및 다항회귀 모델의 예측값 계산

y\_lin\_pred = lr.predict(X)

y\_quad\_pred = pr.predict(X\_quad)

mse\_lin = mean\_squared\_error(y,y\_lin\_pred)

mse\_quad = mean\_squared\_error(y,y\_quad\_pred)

r2\_lin = r2\_score(y,y\_lin\_pred)

r2\_quad = r2\_score(y,y\_quad\_pred)

print('MSE\tLinear: %.2f,\tQuadratic: %.2f' %(mse\_lin,mse\_quad))

print('R2\tLinear: %.2f,\tQuadratic: %.2f' %(r2\_lin,r2\_quad))

plt.scatter(X,y,label='Training Data')

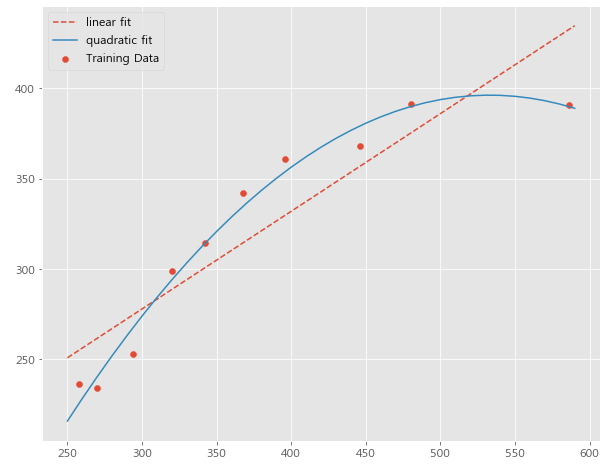
plt.plot(X\_fit,y\_lin\_fit,label='linear fit',linestyle='--')

plt.plot(X\_fit,y\_quad\_fit,label='quadratic fit')

plt.legend(loc=2)

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



한 줄 한 줄 코딩을 해보시고 무슨 의미인지 구글링을 해서 찾아 보시고 위와 같이 그래프도 한 번 그려보시면 좋을 것 같습니다. 앞으로 모델링을 하시려면 numpy 배열을 많이 사용하시게 되는데 다음 시간에는 넘파이 배열에 대해서도 간략히 설명을 드리도록 하겠습니다. 그러면 건강하시고 다음 시간에 뵙겠습니다. 안녕히 계세요.

2019.12.18 (수)

안녕하세요? 송년을 잘 보내고 계신지요? 금년도 얼마 안 남았고 크리스마스도 다가 오고 있습니다. 연말이라서 약속도 많으실거라 생각합니다. 그리고 조만간 미세먼지가 중국으로부터 많이 날아와서 한참동안 힘드실텐데 건강 잘 챙기시고 한 해 마무리 잘 하시길 빕니다.

소그룹 모임이 생기1실만 대상이었는데 이제는 생기2,3실로도 확대가 되어서 저번 주에 추가로 모집하여 총 31명이 되었습니다. 그리고 저번주에 센터장님께도 간단히 보고를 드렸습니다. 잘 부탁한다는 말씀과 함께 여러 부문에서 모였으니 무엇이라도 좋으니 다양하고 많은 시도와 연구가 있기를 원하셨습니다. 또한 실습을 할 때, 소하교육센터를 앞으로 이용하게 될 것과 부족한 것에 대해서는 빅데이터실 지원도 요청할 것이라고 말씀드렸습니다. 이 참에 소그룹이름도 백인택 책임의 전적인 영감으로 ‘슬리데린’으로 지었습니다. 영어로 slytherin은 해리포터 시리즈에 등장하는 호그와트 마법학교의 4개 기숙사 중에 하나입니다. 파이썬 언어의 파이썬도 고대 그리스 신화에 나오는 피톤이라는 뱀이름이기 때문에 뱀을 상징물로 하는 슬리데린으로 짓게 된 것입니다. 뱀은 날렵하고 강하며 목적 달성을 위해 지혜롭고 야심차고 재간이 넘치므로, 그룹명 슬리데린은 앞으로 서로 함께 함에 있어서 뭐든지 매우 빠르고 지혜가 넘치는 그룹이 되길 바란다는 의미가 담겨 있기도 합니다. 또한 앞으로 강의내용 관련이라든지 공지사항 등 전달하고 싶은 메시지는 다음 주소에 남길 것인데요, 주소를 복사하셔서 핸드폰상에서 언제든지 확인하시면 되겠습니다.

* 데이터 분석그룹 알림 페이지 : https://phil-bang.github.io/DATA\_analysis\_group/

멤버 단체 카톡방으로도 여러가지 전달을 할 수가 있지만, 페이지를 하나 만들어서 계속 이력을 남겨서 확인해 가는 것이 전체를 볼 수도 있고 좋아서 만들어 보았습니다. 또한 강의 나눔 중에 중요하다 싶은 내용은 일부 발췌 정리하여 ‘모임진행’란에 링크를 걸어 놓으려 하고 있습니다. 모임이 격주라서 사실 모임 횟수가 많은 것은 아닙니다. 잊어먹을만 하면 만나고 또 잊어 먹을만 하면 만나는 수준이 되지 않을까 하는 걱정도 있긴 합니다. 그래서 앞으로 보내드릴 실습 파일은 집에서도 시간을 내서 해볼 수 있으면 좋을 것 같습니다.

제가 최근에 제 지인과 얘기를 하는 중에 이 뽀개기 파일이 제조부서 누군가에게 보내지고 있다는 것을 알게 됐습니다.^^ 조심스레 물어보더라고요. 제조부서 자신의 지인에게 자료를 계속 공유해도 되냐고 말이죠.^^ 저 모르게 전파하는 사람이 있다는 것을 저는 모르고 있었던거죠. 그래서 기아 직원이면 누구든 괜찮다고 했습니다. 사실 제조부서 그 분에게 오히려 제가 고마운 일이죠. 제조부서 무명인님, 혹시 이 편지 글을 읽고 계시다면 이것만 기억해 주세요. 이 데이터 뽀개기 메일은 어쩌면 저보다 무명인님에게 더 필요할지도 모릅니다. 왜냐하면 공장에서 제일 가까이 있기 때문이죠. 공장이 원유가 나오는 유전이라면 무명인님은 유전 한 복판에 있는 것이고, 저는 유전과는 조금 떨어진 곳에 있는 것과 같습니다. 제조부서에서 일하면서 시간을 내서 공부하기가 남들보다 더 열악하고 도와주는 사람도 없고 정말로 외롭고 어렵겠지만 그럼에도 불구하고 이렇게 관심을 갖고 공부하고자 하는 것에 대해 정말로 감사를 드립니다. 설령 나중에 현업에 지쳐서 공부를 그만둔다고 할지라도 오늘 무명인님의 그 도전은 정말로 값지다는 생각을 하고요, 힘들 미래의 그 때에 오늘 무명인님의 열정을 꼭 생각해 주시기 바랍니다. 공부하시다가 어려운 것이 있으면 언제든지 메일 주시면 모르는 것은 모두 빼고(^^;;) 아는 한 최대한 짜내서 알려 드리겠습니다.^^ 무명인님! 정말로.. 파이팅입니다.^^

다행인 것은 예전 같았으면 이렇게 제가 메일을 보내고 모바일로 공지를 하고 서로 오프라인으로 모이고 하는 것이 많은 제약이 있었을 것 같다는 것입니다. 앞으로 공유문화에 대해서 더 파격적인 시대적 변화를 앞두고 있기 때문에 생각과 판단만 제대로 한다면 서로 공유를 하는 것은 시간문제가 될 것입니다. 그 만큼 공간적인 제약이 점점 줄어든다는 것이죠. 제조부서 무명인님도 그래서 몸은 떨어져 있지만 조만간 더 가깝게 만날 날이 올거라 생각합니다.

인터넷과 모바일 기기가 시간과 공간적인 면에서 사람과 사람을 매우 가깝게 하고 있기 때문에, 이에 따라 사실 우리가 매일하는 업무도 변화를 해야하는 것이 자연스러운 현상입니다. 세상에는 다양한 협업 소프트웨어가 쏟아져 나오는 상황에 있고 자신의 노하우를 오픈소스로 무료로 공개를 하여 또다른 공유 시대의 바람이 불고 있는 상황입니다. 이에 발맞추어 현대자동차 그룹내 변화의 꿈틀거림은 아직은 그 시초가 아닐까 생각을 하게 됩니다. 약간 늦은 감은 있지만 대기업의 큰 덩치를 생각한다면 그리 늦지는 않았다고 생각합니다. 알아도 시도조차 하지 못하는 기업들도 많으니까요. 아래 그림을 잠깐 생각해 보도록 하겠습니다. 스마트워커가 되기 위한 세 가지 관계입니다. ‘일하는 방식의 혁명’이라는 책 안에 있는 그림을 인용해 보았습니다. 책에서 설명하는 내용과는 다를 수 있습니다.

**SMART**

**Worker**

공간과 시간과 업무의 3가지의 관계입니다. 톱니 바퀴로 서로 물려있는 모습입니다. 그리고 스마트 워커가 되기 위해서는 이 톱니 바퀴가 잘 돌아가야 하겠죠. 만약 하나의 바퀴가 천천히 도는데 다른 하나의 바퀴가 빨리 돌려고 한다면 엄청난 스트레스를 받게 되고, 톱니는 멈춰 있거나 아주 천천히 돌게 될 것입니다. 스트레스를 안 받고 잘 돌아가려면 서로간 신뢰가 있어야 하고 서로의 자유도도 높아야 합니다. 물론 윤활류도 뿌려줘야 하겠죠.

카톡을 예로 들어보겠습니다. 카톡은 우리 업무에 공간적이 제약을 넘어서게 해 줌으로서 일의 방식에 큰 변화를 주었죠. 해외 출장을 가서도 보이스톡으로 언제든지 공짜로 연락이 가능하고 실시간으로 정보를 전달할 수가 있게 되었습니다. 그렇다면 공간 톱니가 잘 돌고 업무 톱니도 잘 돌아가는데 시간 톱니는 어떨까요? 시간에 대한 자유에 많은 제약이 가해지고 있다는 것을 모두 인정할 수밖에 없을 것입니다. 한 톱니 바퀴가 다른 톱니 바퀴에 맞물려서 영향을 주기 때문이죠. 스마트 워커를 만들어 내려면 어떻게 해야할까요? 시간의 효율성을 높여야 하겠죠. 근무 형태에 점점 변화가 와야하고 유연한 협업 체제가 점점 더 필요해 집니다. 근무 방식이 전혀 바뀌지 않고서는 톱니바퀴에 강한 스트레스가 걸릴 수 있습니다. 효율적인 근무 방식이 무엇인지 많은 변화 시도와 연구가 필요하죠. 그나마 사무실에서 근무하는 직원들에 대해서는 톱니바퀴가 돌아는 가는데 현재 현대기아 현장에 대해서는 저 톱니바퀴는 멈춰 있다고 보시면 되겠습니다. 공간 톱니바퀴는 돌아가려고 애를 쓰고 있지만 시간과 업무 톱니바퀴는 전혀 움직이지 않고 있다고 보시면 되겠습니다.

또한 우리가 일하는 업무 현장에는 너무나 많은 앱들이 따로따로 분리가 되어 움직이고 있습니다. M채널과 카톡의 분리는 대표적인 비효율적인 업무형태죠. 물론 기아에서 이것을 연결시키려는 시도도 있었던 것으로 기억은 합니다. 사실 업무가 앰채널보다 카톡으로 더 많이 공유되고 있고 카카오톡은 같은 회사가 아니기 때문에 통합을 시도하기가 매우 어려운 상황이긴 하죠. 업무는 모바일을 통해 자료도 자유자재로 다니고 영상회의도 하고 채팅이나 댓글기능 등 통합적인 협업 관계를 원하는데 회사는 보안 등 많은 제약으로 인해 앰채널 수준을 아직 넘지 못하고 있는 것이 현실입니다.

요컨대, 결론적으로 데이터 분석은 기존 협업 업무의 틀을 깨고 톱니 바퀴를 움직이게 하는 또 다른 스마트 워크의 도구가 될 수 있다는 것입니다. 공간 톱니 바퀴가 문제가 된다면 사실 공간 자체보다 시간과 우리 업무 자체에 문제일 수도 있습니다. 우리 업무와 시간에 데이터 분석문화라는 새로운 패러다임을 접목시킨다면 톱니 바퀴는 빠르지는 않겠지만 다시 시동이 걸릴 수 있을지도 모릅니다. 데이터를 기반으로 하고 있기 때문에 업무의 판단,정확성을 높일 수 있고, 시간을 절약할 수 있고, 사실적인 데이터를 기반으로 대화가 이뤄지기 때문에 수평적인 소통을 확대할 수 있고, 새로운 인사이트를 발견해 나감으로서 업무 창의성을 높일 수 있다는 점에서 데이터 분석은 앞으로 스마트워크의 핵심이 될 것입니다. 물론 첫 술에 배부를 수가 없기 때문에 열정적이고 재미있게 공부하는 다양한 시도들이 곳곳에서 일어나야 하겠죠. 제 편지도 그 수많은 시도들 중에 작은 하나의 시도라 볼 수 있습니다. 아마도 다음 시도의 주체는 이 편지를 읽고 있는 여러분들이 되어 주셔야 합니다. 그 동안 지독하게도 따라 다니는(특히 한국에는) 연령,학연,지연,경력 등의 꼬리표와 전혀 상관없이 모든 분들이 시도하실 수 있고 가능합니다. 제 2의 새로운 삶을 살고 싶으신 마음을 가진 분이라면 누구가 가능합니다. 방법과 길이 따로 있는 것은 아닙니다. 지금 있는 위치에서 지금까지 생각해온 방식이 아닌 데이터 분석문화 정착을 위해 생각만 살짝 바꾸셔도 이미 여러분만의 새로운 도전은 시작된 것입니다. 그럼 오늘도 파이팅입니다.^^

오늘도 서두가 너무 길었는데 그럼 오늘의 할 일을 해야죠.^^ 저번 주에는 매우 중요한 회귀분석의 원리에 대해서 알아보고 마지막에 예제를 드리고 마쳤는데요. 오늘은 그 예제를 하기 위해서 필요한 배열을 만들어 주는 numpy 라이브러리에 대해서 알아보고 예제 설명에 들어가 보겠습니다.

현재 X,y값에 각각 10개 데이터를 입력하여 대괄호([],square bracket)로 감쏴 줬는데, 대괄호로 감쏴서 만든 데이터의 타입이 list(리스트)라고 합니다. 이 상태에서 array로 변경해 주면 1차원 array(배열)이 되는 것입니다. 아래 코드를 입력해서 속성을 한 번 보겠습니다.

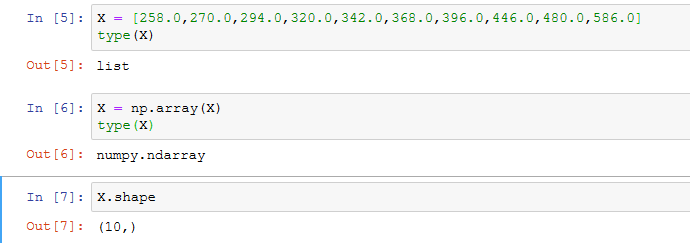
X = [258.0,270.0,294.0,320.0,342.0,368.0,396.0,446.0,480.0,586.0]

type(X)

X = np.array(X)

type(X)

X.shape



shape은 행과 열의 구조를 보여줍니다. ndarray는 n-dimensional array의 약자로 n개의 차원으로 이뤄진 배열을 뜻합니다. Shape을 한 결과가 (10,)가 되었으니 1차원 배열이 됩니다. 행은 있는데 아직 열은 없으니 2차원 행렬은 아닌 것이죠. 대괄호가 하나 있는 배열은 1차원 배열이라고 합니다. 그런데 왜 배열을 사용할까요? 아래 그림을 보겠습니다.

4

3

2

1

리스트 : [1,2,3,4] →

4

3

2

1

배열 : array([1,2,3,4]) →

위에서 보듯이 리스트는 각 요소가 하나씩 방을 따로 차지하고 있는데, 배열은 방 하나에 4개가 들어있는 것입니다. 그러면 데이터를 불러올 경우에 리스트는 각 방에 들어가서 하나씩 불러와야 하는데, 배열은 방에 들어가서 4개를 한꺼번에 불러올 수 있는 편리함이 있죠. 그러니까 배열이 더 속도도 빠르고 용량도 덜 차지하게 되는 것입니다. 그래서 복잡한 모델링을 할 경우에 배열을 이용하는 것이죠. 그러면 차원은 무엇일까요?

차원은 배열 방의 구조라고 보시면 됩니다.

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

1

2

3

4

우측 첫번째 배열모습은 하나의 방에 한 방향으로 되어 있어서 1차원이고,

그 아래 모습은 방이 하나인데 방향이 2가지 방향이 있네요. 그래서 2차원입니다.

맨 아래 모습은 방이 하나인데 방향이 3가지 방향이 있습니다. 3차원이죠.

이런 식으로 배열이 차원으로 표현이 되는데, 차원이 많아지면 많아질수록

데이터량은 더 많아지고 복잡해지겠죠. 마치 아파트에 번지수별로 사람이 들어가서

사는 것처럼 배열도 표현하는 구조가 있고 번지수가 다 있습니다.

그런데 리스트처럼 따로따로 사는 각 방을 쓰는 것이 아니라,

한 방을 쓰는 한 가족이죠. 차원이 많아지면 다양한 모양의 아파트가 될 텐데

그 아파트 하나가 방 하나라고 보시면 되겠습니다. 아파트가 복잡하면 필요에

따라서 아파트를 분리해야 하는 경우도 있겠죠. 그러면 어떻게 표현하는지 보겠습니다.

array([1,2,3,4]) 대괄호가 1개가 있으면 1차원이고,

array([[1,2,3,4],[1,2,3,4]]) 대괄호가 2개가 있으면 2차원입니다.

2차원은 여기에서 바깥쪽 대괄호는 행을 말하고, 안쪽 대괄호는 열을 말합니다. 그러면 바깥쪽 대괄호를 기준으로 볼 때, 안쪽 대괄호는 2개가 있으니까 2행, 안쪽 대괄호만을 볼 때, 개수가 4개가 있으니까 4열이 됩니다. 그래서 2행 X 4열 구조의 2차원 배열이 됩니다. 물론 대괄호가 3개가 되면 당연히 3차원이 되겠죠.

Array([[[1,2,3,4],

[[1,2,3,4]]])

맨 바깥쪽 대괄호를 기준으로 볼 때, 두번째 대괄호 개수가 2개니까, 2행이 되고, 두번째 대괄호를 기준으로 각각 1열씩 있고, 마지막으로 맨 안쪽 대괄호를 기준으로 4개가 있으니까 깊이가 4가 됩니다. 그래서 이것은 2행 X 1열 X 4깊이의 구조를 가진 3차원 배열이 되겠습니다. 그럼 리스트로 계산하는 것과 배열로 계산하는 것과 어떤 것이 더 빠를지 한 번 보겠습니다. data에 2를 곱하는 반복문입니다. 리스트는 방이 분리가 되어 있기 때문에 값을 하나하나 불러와야 해서 아래와 같은 for문을 작성해야 합니다.

data = [1,2,3,4]

a = [ ]

for i in data:

a.append(2\*i)

a

결과] [2,4,6,8]

그런데 array은 방이 하나이기 때문에 아래와 같이 한 번에 작업이 됩니다. 바로 연산이 가능하다는 것이죠.

x = np.array(data)

2 \* x

결과] array([2,4,6,8])

그러면 우리가 저번 주에 했던 예제로 돌아가 보도록 하겠습니다. 먼저 첫번째 X,y데이터 입력부터 보겠습니다.

X = np.array([258.0,270.0,294.0,320.0,342.0,

368.0,396.0,446.0,480.0,586.0])[:,np.newaxis]

y = np.array([236.4,234.4,252.8,298.6,314.2,

342.2,360.8,368.0,391.2,390.8])

X배열값을 만드는데 리스트가 한 개 들어 있네요. 그러면 1차원이 되는데, 그 뒤에 보니까 뭐가 하나 붙어 있네요?

[:,np.newaxis]라는 건데요. 대괄호는 [행,열]을 의미하는데, 행은 콜론이 왔으니까 전체를 말하는 것이고, 열에 np.newaxis가 왔기 때문에 열에 새로운 축을 부여하라는 뜻이 됩니다. 아까 차원 얘기할 때, 방향이 하나, 두 개, 세 개 등 이런 방향의 개수로 차원을 표현했습니다. 그 방향이 영어로 하면 axis가 되겠습니다. 벡터를 말하는 것이죠. x축,y축,z축 등 이런 방향이 있는 것을 말합니다. 그러니까 결국 newaxis는 방향을 새롭게 한 개 부여함으로써 차원이 하나 늘어나게 되는 것을 말합니다. X값이 1차원이었는데 축이 열방향으로 늘어나게 됨으로서 2차원이 되는 것이죠. y값은 그냥 그대로 1차원 배열입니다. 회귀분석에서 X값과 y값을 가지고 모델을 만드는데, 그 X,y값은 배열값을 사용하는 것이 기본이고, 1차식이 아니라 2차 다항식으로 표현을 하려고 하기 때문에, X값은 1차원이 아닌 2차원 배열값이 필요하게 됩니다. 그래서 2차원으로 표현했고, y값은 2차식 설명변수에 대해서 하나의 값들을 가지기 때문에 1차원만 해도 되겠습니다. 물론 둘 다 반드시 배열값은 되어야 합니다. 모든 머신러닝은 배열을 기본으로 사용합니다.

예제 코딩이 어떻 방식과 순서로 만들어 졌는지 아래와 같이 정리해 보았습니다.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[모델생성 훈련용 데이터 X,y값에 대한 선형회귀 1개, 다항회귀식 1개 만드는 과정]

1. X,y 데이터값을 만든다. (array로 만든다)

2. X,y값을 가지고 lr모델을 만든다. (모델은 실제값을 가지고 만든다)

* 이렇게 만든 lr모델을 가지고 실제 X값이 아니라, X값과 동일한 범위에서 임의의 값 10개 선택
* 선택한 10개 X값을 X\_fit로 놓고, lr모델에 대입하여 10개 피팅값 y를 구하고 y\_lin\_fit으로 놓기

→ X\_fit, y\_lin\_fit

3. 다항식 변형함수를 선언하고 X값을 함수모델에 대입하여 2차 다항식 값으로 변형 -> X\_quad

4. X\_quad,y값을 가지고 pr모델을 만든다. (모델은 실제값을 가지고 만든다)

* X\_fit값을 가지고 2차 다항식의 X값을 만든다. → X\_quad\_fit
* X\_quad\_fit값을 pr모델에 대입하여 피팅값 y를 구한다. → X\_quad\_fit, y\_quad\_fit

5. lr과 pr모델 평가를 한다.

* lr모델 : 실제값 X를 대입한 예측값 y\_lin\_pred와 y를 비교한다.
* pr모델 : 실제값 X\_quad를 대입한 예측값 y\_quad\_pred와 y를 비교한다.
* MSE와 R^2결정계수값을 구한다.

6. 회귀선 시각화

* 단순선형회귀: X\_fit값, y\_lin\_fit값
* 다항회귀 : X\_fit값, y\_quad\_fit값

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

지금까지 배열(array)에 대해서 설명 드린 부분이 위 1번 사항이 되겠습니다. 그럼 2번부터 설명을 드리도록 하겠습니다. 우선 모델을 2개를 만드려고 합니다. 선형회귀모델과 2차 다항식모델, 이렇게 2개의 모델을 만들고 시각화도 해 보도록 하겠습니다.

lr = LinearRegression()

pr = LinearRegression()

lr모델이 선형회귀선을 만들어 주는 모델이 되겠고, pr모델이 2차 다항식곡선을 그려주는 모델이 되겠습니다. 현재는 빈 알고리즘(정의된 함수)입니다. sklearn이라는 라이브러리에 LinearRegression()이라는 함수가 있어서 이 함수를 이용해서 값을 대입하면 회귀선을 그릴 수 있는 값을 만들 수가 있게 됩니다.

그럼 단순선형회귀선을 그리기 위한 값을 생성해 보도록 하겠습니다. X,y값은 실제값이고 선형회귀선을 그리는 X,y값을 fitting값이라고 합니다. 그래서 X\_fit, y\_fit라고 표현을 하겠습니다.

*실제값(실험결과값)*

*선형회귀선*

*회귀값(피팅값)*

X값과 X\_fit값은 사실 동일한데 회귀선을 조금 더 길게 그리기 위해서는 범위가 약간 더 넓어야 해서 편의상 X\_fit값을 다시 정해서 만들어 보도록 하겠습니다. 다음 코드를 입력하고 실행하겠습니다.

# X값 범위와 동일한 범위로 X\_fit값을 임의로 10개를 만들어 2차원 array로 변경해 준다.

X\_fit = np.arange(250,600,10)[:,np.newaxis]

# X와 y값을 입력하여 선형회귀 fitting하여 모델을 만든다

lr.fit(X,y)

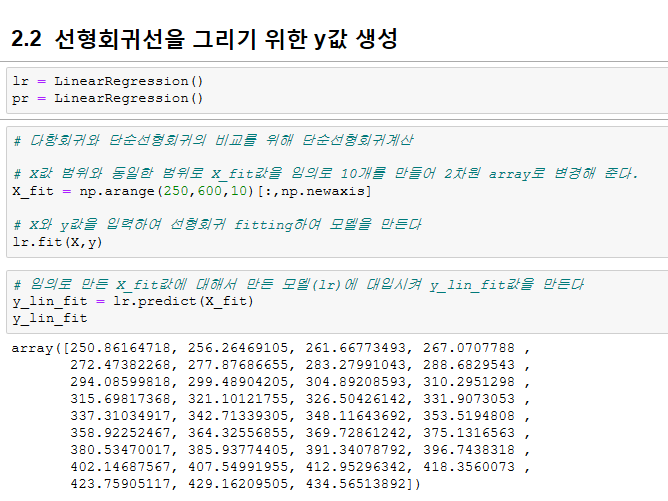
X값의 최소값이 258이고 최대값이 586이니까, 넉넉하게 범위를 잡아서 250~600으로 해서 X\_fit값을 만들었습니다. 그리고 lr 모델을 만들어주는 함수(알고리즘)에 X,y실제값을 대입시켜서 모델을 자동으로 생성했습니다. 간단하죠? 그리고 저 모델을 가지고 회귀값을 만드는 것입니다.

# 만든 X\_fit값에 대해서 만든 모델(lr)에 대입시켜 y\_lin\_fit값을 만든다

y\_lin\_fit = lr.predict(X\_fit)

y\_lin\_fit

선형회귀니까 y\_fit 대신에 y\_lin\_fit으로 했습니다. X\_fit값을 lr모델에 대입해서 y\_lin\_fit값을 생성했습니다. 그렇게 해서 X\_fit와 y\_lin\_fit값을 가지고 직선을 그리면 이것이 선형회귀선이 되는 것입니다. 간단합니다.



단순선형회귀식은 만들었으니까, 다음에는 2차식 다항회귀식을 만들어 보겠습니다. 다항식은 X에 관한 항이 하나가 아니라 2개이기 때문에 상수항,1차항,2차항을 분리해서 데이터를 만들어 줘야 합니다. 그러니까 X값을 2차 다항식에 맞게 변형을 시켜줘야 한다는 것이죠. X\_fit값을 가지고 2차 다항식에 맞게 데이터를 변형시켜줘서 그 값을 X\_quad라고 하겠습니다. Quad는 quadratic의 약자로 2차식이라는 뜻을 가지고 있습니다.

: *2차 다항식*

X값만 2차식으로 변형시키고 y값은 그대로 사용하면 되겠습니다. 다음 코드를 입력/실행하겠습니다.

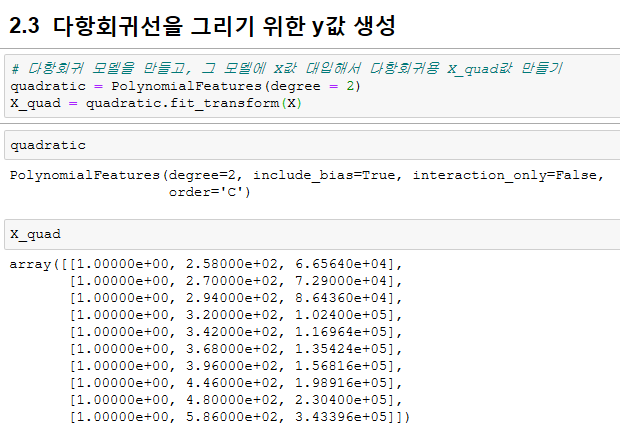
# 다항회귀 모델을 만들고, 그 모델에 X값 대입해서 다항회귀용 X\_quad값 만들기

quadratic = PolynomialFeatures(degree = 2)

X\_quad = quadratic.fit\_transform(X)

PolynomialFeatures()함수의 polynomial은 다항식이라는 뜻이고 Features는 속성을 말합니다. 그러니까 다항식의 속성으로 바꿔주는 함수가 되는 것이죠. 그 함수 안에 인수를 보면 degree가 있습니다. 2로 되어 있으니까 2차식이 되겠습니다. 물론 3차, 4차식으로도 바꿀 수 있겠죠. 그렇게 만든 quadratic 변형 알고리즘에 X값을 대입하여 X\_quad값을 만들었습니다. 그런데 X\_fit값이 아니라, 왜 X값을 대입했을까요? 사실 X\_fit에는 X가 아닌 다른 값도 포함되어 있기 때문입니다. 단순히 직선을 길게 하기 위해서 우리가 강제로 범위를 넓혀서 만들었기 때문에 그래프를 그리기 위한 값이지 실제값은 아니죠. 그래서 실제값 X를 대입해야 실제값 X\_quad가 생성되는 것입니다.

그러면 X\_quad값을 한 번 볼까요?



2차식 곡선을 그릴 수 있는 X에 관한 값으로 배열이 생성되었네요. 2차원 배열이고요, 10행 3열의 배열이 되겠습니다. 그러면 X\_quad와 y값에 대한 2차식 다항회귀 모델을 만들어 보겠습니다.

# X\_quad와 y에 대한 다항회귀모델 만들기

pr.fit(X\_quad,y)

이렇게 만들어진 모델 pr에 대해서 좀 더 범위가 넓은 곡선을 그리기 위해서 아까 사용했던 X\_fit값을 사용해서 피팅용 y값을 생성해 보도록 하겠습니다. 그런데 X\_fit값은 2차 다항식용으로 변형을 안 시켰기 때문에 변형을 먼저 시켜주고 모델에 대입을 하겠습니다.

# X\_fit값을 다항회귀모델 fitting용으로 값을 변형시키기

X\_quad\_fit = quadratic.fit\_transform(X\_fit)

# 다항회귀모델용으로 변형된 X\_quad\_fit값에 대한 y값인 y\_quad\_fit값을 계산

y\_quad\_fit = pr.predict(X\_quad\_fit)

y\_quad\_fit

이제 2차 다항회귀곡선을 그릴 수 있는 X\_quad\_fit과 y\_quad\_fit을 잘 생성했습니다. 이것으로 그래프를 그리면 되는 것이죠. 2 개의 그래프를 동시에 그리도록 하겠습니다.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fig, ax = plt.subplots(figsize=(10,8))

plt.scatter(X,y,label='Training Data (초기 실제값: 훈련용)') # 초기 실제값

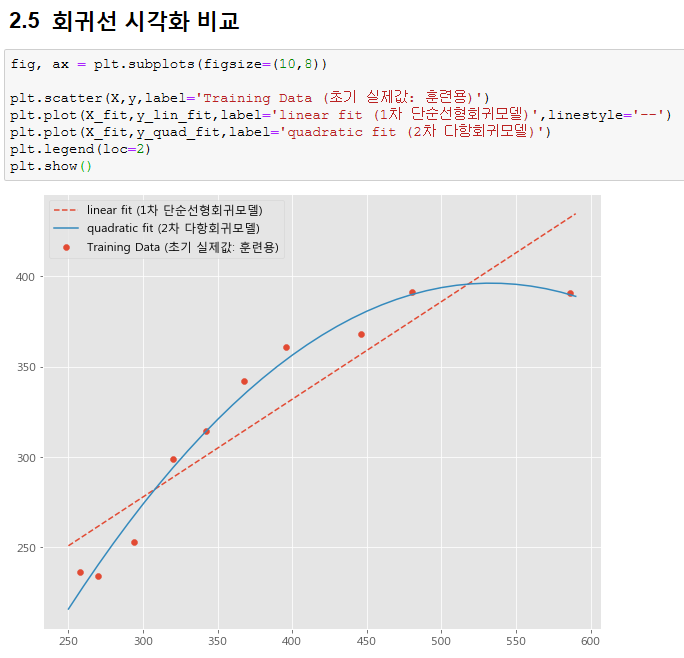
plt.plot(X\_fit,y\_lin\_fit,label='linear fit (1차 단순선형회귀모델)',linestyle='--') # lr모델로 fitting한 값

plt.plot(X\_fit,y\_quad\_fit,label='quadratic fit (2차 다항회귀모델)') # pr모델로 fitting한 값

plt.legend(loc=2)

plt.show()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



그래프 2개가 잘 그려졌습니다. 파란선은 2차 다항식 회귀선이고 빨간 점선은 1차 회귀선을 나타내고 있습니다. 어떤 회귀선이 실제값을 잘 나타내 준다고 생각하시나요? 그것은 아마도 곡선일 것입니다. 그렇다고 회귀직선이 아주 터무니없이 그려진 것은 아닙니다. 이렇게 2 가지 모델로 회귀선을 그려봤는데, 이것이 잘 그려졌는지 아닌지 평가하는 방법이 있다고 했죠? 오차제곱평균값이 기억나시나요?

MSE =

그리고 결정계수가 생각나시나요?

MSE값이 작을수록 회귀선 모델이 잘 만들어진 것이고요, 결정계수값이 1에 가까울수록 잘 만들어진 모델이라고 할 수 있습니다. 그럼 평가를 해보겠습니다. 모델이 lr,pr이니까 실제값을 넣어서 y값을 만들어보겠습니다. 그리고 그렇게 만들어진 y값들을 lr모델에서 나온 값은 y\_lin\_pred로 하고, pr모델에서 나온 값은 y\_quad\_pred로 놓겠습니다. 이렇게 나온 y값들이 초기 실제 y값과 비교하면 되겠습니다.

# 단순회귀 및 다항회귀 모델의 예측값 계산

y\_lin\_pred = lr.predict(X)

y\_quad\_pred = pr.predict(X\_quad)

print(y\_lin\_pred)

print('\n--------------\n')

print(y\_quad\_pred)

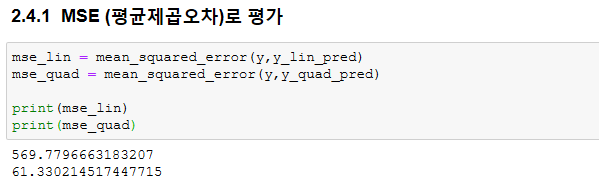
평균제곱오차값을 구할 수 있는 함수가 있기 때문에 비교할 값만 넣으면 되겠습니다. 아래와 같이 실제값과 모델에 대입해서 나온 값을 차례로 입력하면 됩니다.

mse\_lin = mean\_squared\_error(y,y\_lin\_pred)

mse\_quad = mean\_squared\_error(y,y\_quad\_pred)

print(mse\_lin)

print(mse\_quad)

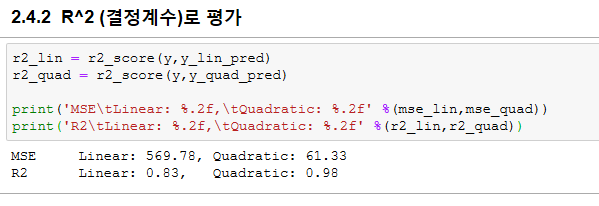


r2\_lin = r2\_score(y,y\_lin\_pred)

r2\_quad = r2\_score(y,y\_quad\_pred)

print('MSE\tLinear: %.2f,\tQuadratic: %.2f' %(mse\_lin,mse\_quad))

print('R2\tLinear: %.2f,\tQuadratic: %.2f' %(r2\_lin,r2\_quad))



결과가 나왔습니다. 직선회귀보다 곡선회귀가 MSE값이 더 작네요. 결정계수를 보니까 역시 곡선회귀가 더 1에 가까우니까 더 정확하게 모델링이 된 것입니다. 하지만 결정계수가 0.65이상이면 일반적으로 모델이 잘 나온 것으로 평가가 되기 때문에 직선회귀모델도 정확도가 떨어진다고 볼 수는 없습니다. 실제값의 양이 더 많다면 그 모델간 평가값 차이가 확연히 드러나겠는데 데이터 양이 적기 때문에 차이가 없어 보이는 것입니다.

데이터를 알고리즘에 넣고 트레이닝시켜서 모델을 만드는 모델링의 이유는 앞으로 도래할 미지의 값에 대한 예측을 하기 위한 것입니다. 과거의 값을 훈련시켜 미래의 값을 예측하고 또 평가를 계속해 나간다면 어떤 조건이 크게 변하지 않는 상황에서는 이 모델이 미래를 예측해 준다고 볼 수가 있는 것입니다. 앞으로 무슨 값이 들어와도 이 모델 안에서 크게 이탈되지 않을 것이기 때문이죠. 그래서 모델링과 평가를 계속 반복하면 엄청난 정확도를 가진 예측파워를 발휘하기 때문에 이 모델에게 사람이 일을 맡겨도 되는 것입니다. 그것이 우리가 흔히 말하는 인공지능이 되겠습니다. 그러나 이 인공지능에 대해 만능이라는 너무나 큰 환상을 갖게 되면 안됩니다. 그리고 도깨비 방망이 같이 뚝딱 모델이 만들어지는 것도 아닙니다. 분명 모델이 만들어지기까지 데이터에 대한 사람의 연구가 계속 이뤄져야 하고 평가해야 하고 다시 업그레이드해야 하는 연속적인 행위가 필요하기 때문에 사실 인공지능 뒤에는 그것을 가능케하는 사람이 있다는 것을 절대로 잊어서는 안됩니다.

그럼 오늘 여기까지 하고 다음 시간에 5편으로 새롭게 찾아뵙도록 하겠습니다. 아마도 내년 1월초가 될 것 같은데 연말 정리 잘 하시고 다가오는 크리스마스도 즐겁게 보내시길 빌겠습니다. 또한 2020년 새해가 다가오는데 새해에는 댁내 평안하시길 빌며 하고자 하시는 일들이 모두 잘 되시길 기원하겠습니다. 감사합니다.