**Data Mining**

**Unsupervised learning** - data is unlabeled. 가지고 있는 데이터에서 분류나 패턴을 찾아냄.

**Machine Learning**

**Supervised learning** – data is labeled. Train data로 훈련후 test data로 검증하며 하는 거.

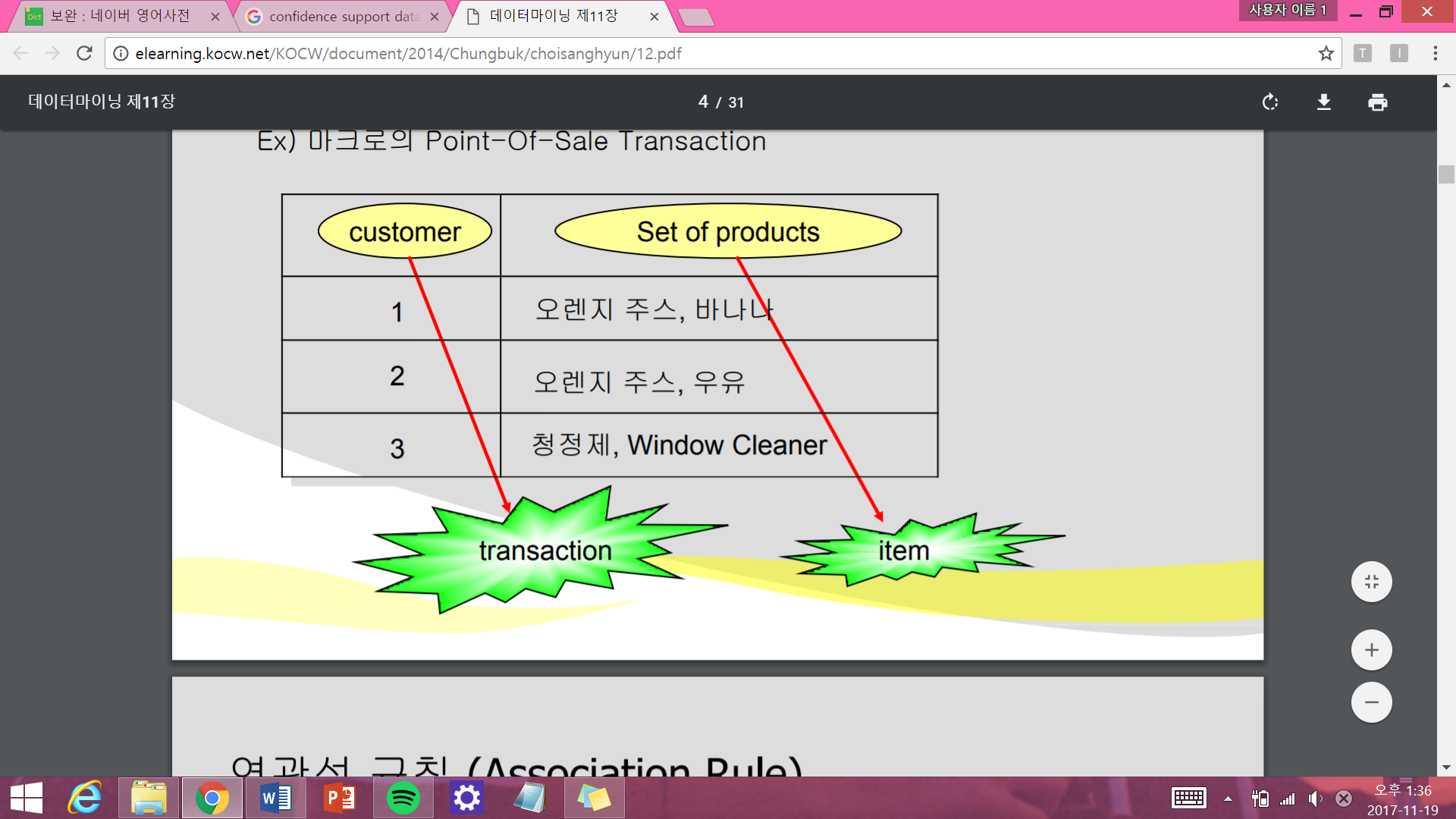
Labeled data – 위도 경도로 온도를 warm hot 이렇게 분류해 놓은 데이터.

위도 경도는 feature. 그래서 나온 온도는 label. 트레이닝 데이터로 분류하는 훈련을 시키고 그 모델로 테스트함. feature들로 유추하는 model

Unlabeled data – feature만 있음

**Unsupervised Learning –** unlabeled data들로 내제된 패턴이나 구조를 찾음

1. Frequsent Itemset – 자주 일어나는 것들
   1. Transaction – 사건
   2. Item – 사건에서 일어난 거



1. Association Rules -support(지지도), confidence(신뢰도), Lift(향상도)

X -> Y : x가 y에 영향을 미친다. 역이 성립하지 않는다

* 1. Support : proportion of total 전체중에 일어난 빈도 수

P(X&Y) =

\*support threshold(문턱) : 빈도에 대한 기준

* 1. Confidence – x가 일어날 경우, 얼마나 y가 발생하는 지 확률.

= x가 의 항목 중에 y가 포함된 빈도

P(X&Y)/ P(X) =

\*Drawback : if x is just too popular and has nothing to do with y actually, the confidence is hard to be considered correct.

보완한 게

* 1. Lift – Y가 그냥(randomly) 구매되는 거에 비해, 관계가 고려되서 구매되는 비율. popularity 고려한 것. Association이 예측하는 능력이 얼마나 향상이 되었나.

**Lift =** \*Support – 전체중에 일어날 확률

Lift > 1 x와 y가 함께 나타날 확률이 **높다**

Lift = 1 둘이 관계가 전혀 없다. 서로 독립적이다. =.

Lift < 1 x와 y가 함께 나타날 확률이 **낮다**

Clustering – 앞에 frequent data set에서 수치를 나타냈다면 얘는 큰 그룹으로 나눠 어느 그룹에 속하는지 분류 cluster = group

**supervised Learning** – labeled data를 이용해 train해서 그걸로 예측한 값을 출력

Regression(회귀) – 출력으로 수치를 내는 것

 Classification – 어디에 속하는 지 분류만

Input

Output

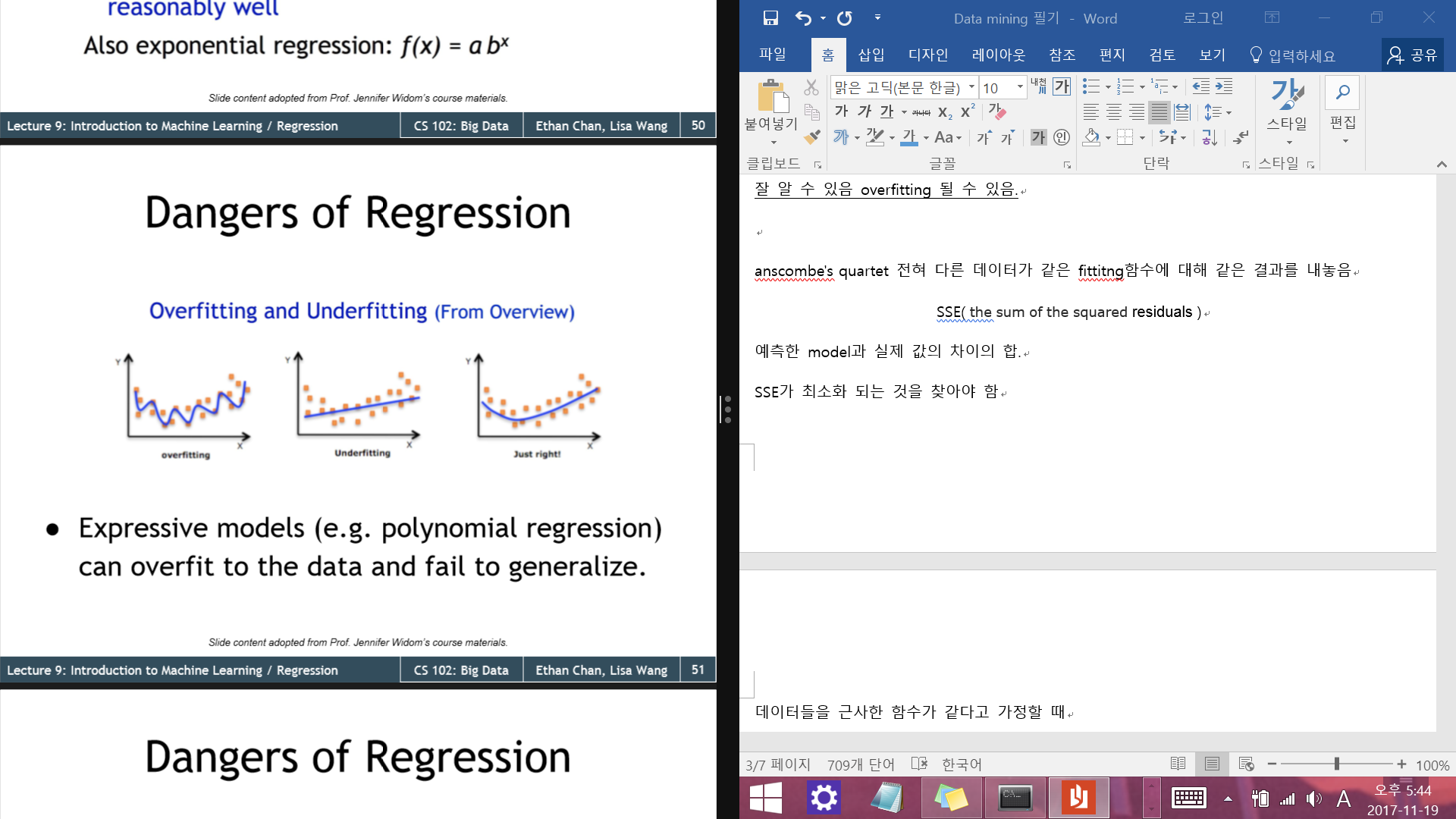
Model (function)

Provided data로 model 만드는 training 후 그걸 이용해(**fitting**) new data로 예측한 값을 내 놓음.

Training data = provided data

그 new data와 예측한 값을 비교하는 것이 **correlation**

1. Fitting



Least Square – 가능하면 작은 차수의 그래프로 찾음

차수가 올라가면 overfitting이 됨

1. SSE

SSE( the sum of the squared residuals ) : 예측한 function과 실제 데이터 값의 차이의 합.

SSE가 최소화 되는 것을 찾아야 함

anscombe's quartet - 전혀 다른 데이터가 같은 fittitng함수에 대해 같은 결과를 내놓음.

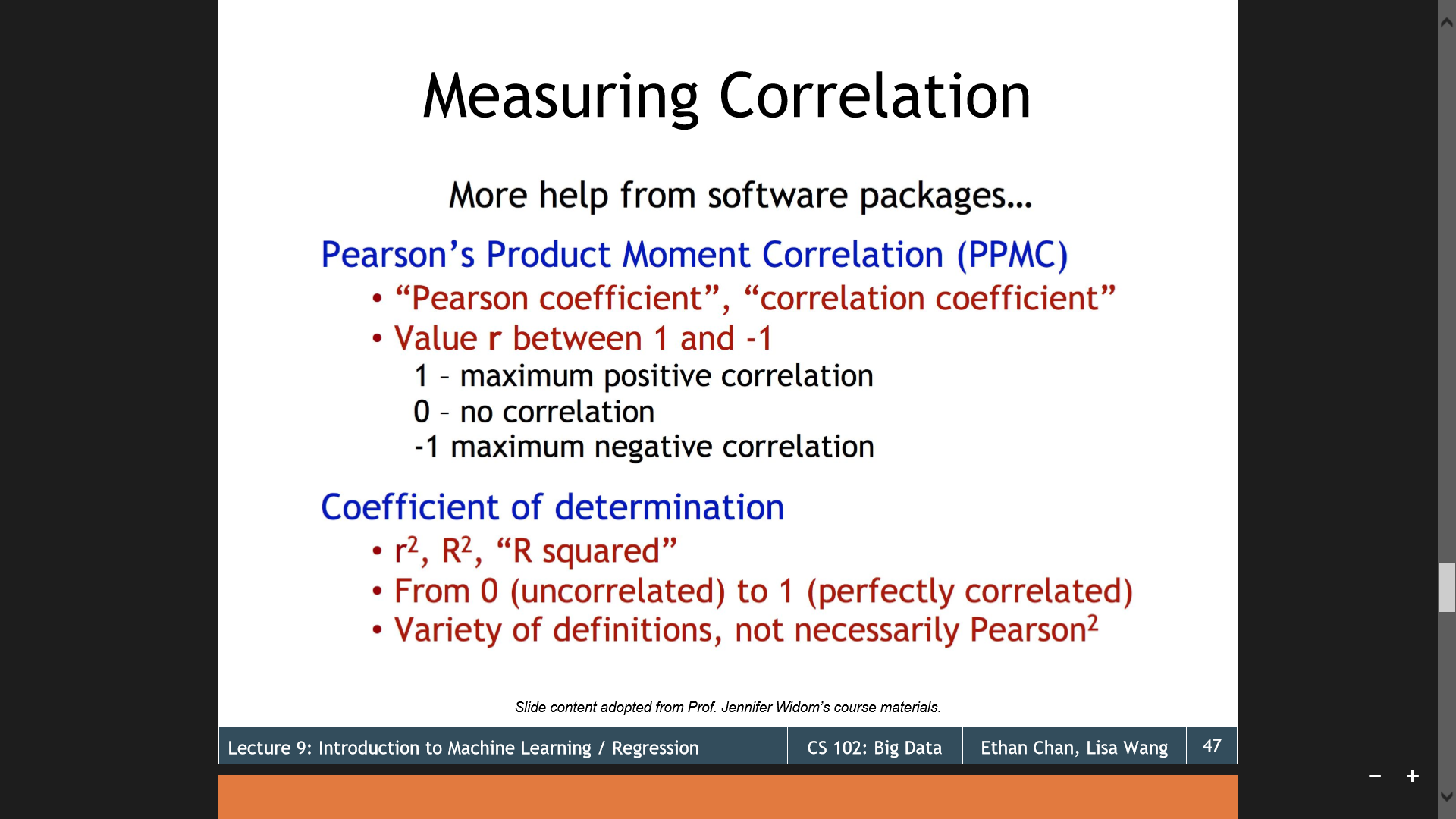
데이터들을 근사한 함수가 같다고 가정할 때

각 데이터 값에서 /함수에 차이를 더해보니 (SSE) 4개의 그래프가 다 같음

->측정하는 방법 (matric)의 맹점

1. Correlation 상관관계

상관 계수는 공분산(COV(X,Y))을 각각의 표준 편차로 나눈값.



1이면 두개가 함께 일어날 확률이 높고

0이면 아예 상관이 없고

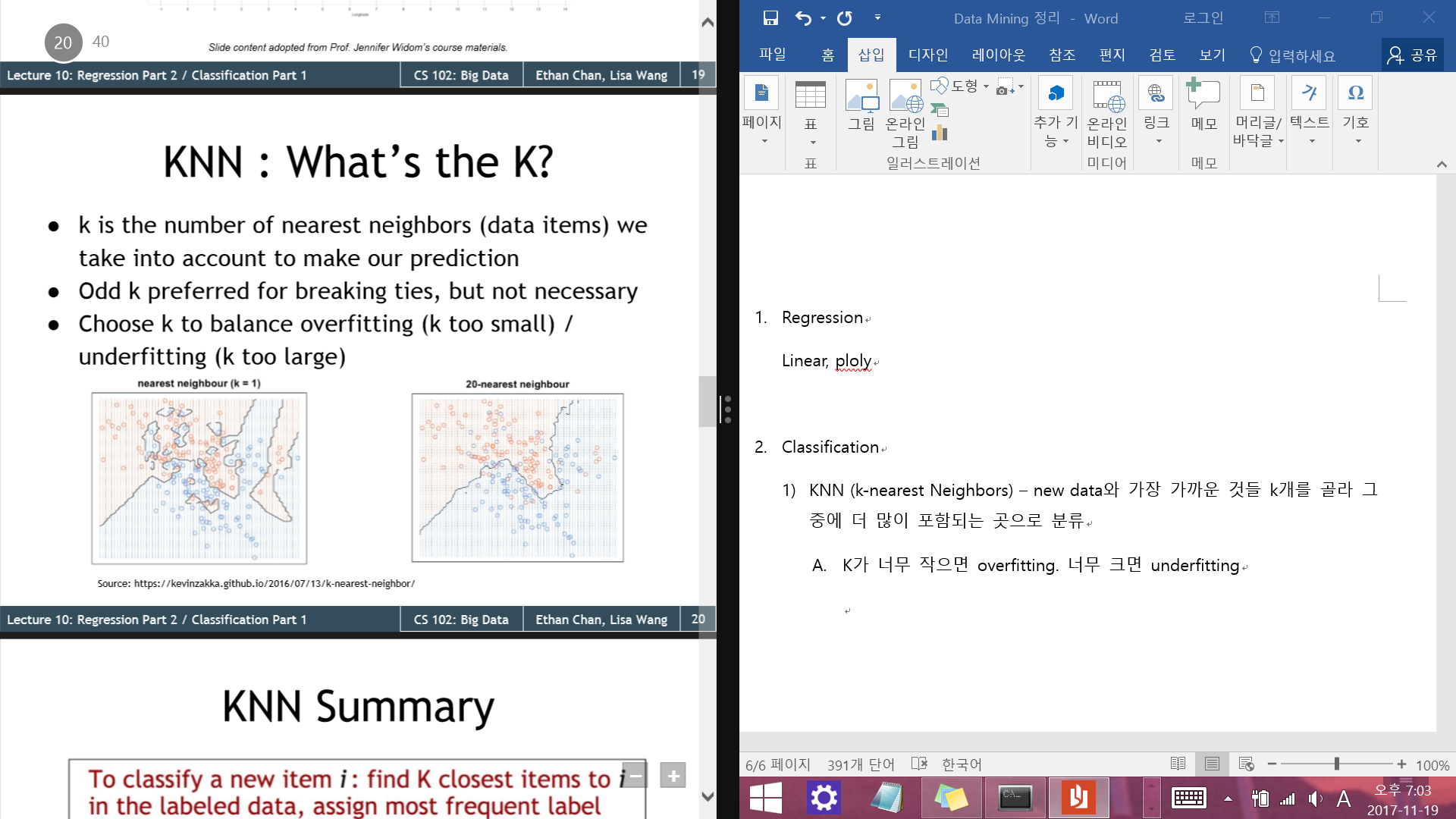
-1이면 두개가 같이 일어나지 않을 확률이 높다

Lift 같은 거

1. Regression

Linear, ploly

1. Classification
2. KNN (k-nearest Neighbors) – new data와 가장 가까운 것들 k개를 골라 그 중에 더 많이 포함되는 곳으로 분류
   1. K가 너무 작으면 overfitting. 너무 크면 underfitting



* 1. 단점 : 범위가 커지면 가까운 것들을 찾는데 시간 오래 걸리고 힘듬

고려해야할 항목이 많으면(변수가 많으면. X1 x2 x3) 효율적이지 못하다.

1. Decision Tree

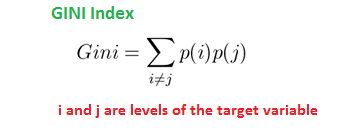
* 조건을 하나씩 주면서 분류하는 방법
* www.r2d3.us/visual-intro-to-machine-learning-part-1/

1. 정확도를 너무 높이면 overfitting의 위험이 있다,
2. The best split – 나누는 기준에 대한 최선의 선택 방법

**Gini index, cross entropy**

1. Gini Index : 섞인 정도가 얼마나 큰지 (mesure of inpurity)

숫자 작으면 지배적인 것들이 많다 -> 나누기 쉽다.



i j 는 label. p=부분/전체(확률)

pi와 pj가 얼마나 섞여 있는 지

P(Target=1).P(Target=0)  + P(Target=0).P(Target=1)   =

1 – P2(Target=0) – P2(Target=1)

I와 j가 독립적이라면, 전체에서 j 확률만 일어날 수와 i만 일어날 수를 빼면, 섞인 정도가 됨.

P0과 p1이 동등하게 나뉘어져 있으면 gini index가 maximum = 1 - 1/k

실제 사용 예 : <https://www.researchgate.net/post/How_to_compute_impurity_using_Gini_Index>

1. 이 전체 데이터에 대한 gini 값을 구하고 before로 둔다
2. Threshold 를 x로 정하고 그에 따라 나뉜 데이터 셋 두개의 gini index를 각자 구한다
3. 그 gini 값에 weight(전체중에 이 데이터 셋이 차지하는 비중(개수))를 곱하면 이 threshold값에 대한 gini index의 값이다
4. X를 바꿔가며 계산하고 before값과의 차이를 계산 (information gain) 값이 가장 큰 것이(x에 따른 gin값이 가장 작은 것) 최선의 gini index
5. Corss Entropy : 무질서에 대한 측정 (지니와 같음. 값이 적을수록 좋음)

<https://stackoverflow.com/questions/1859554/what-is-entropy-and-information-gain>

**로그 사용 이유**

정보량 = h(x) = -log P(x)

브라질이 이길 확률 (P) 은 대략 99%라고 하자. (축구인의 관점에서 볼 때 reasonable 하다)

그렇다면 정보량은, 즉 놀람의 정도는 아래와 같다.

* 브라질이 이기는 경우: -log P(x) = -log(0.99) = 0.01
* 중국이 이기는 경우: -log P(x) = -log(0.01) = 4.6

**[출처]** [Cross Entropy 란 무엇인가 (information theory)](http://blog.naver.com/gyrbsdl18/221013188633)|**작성자** [박효균](http://blog.naver.com/gyrbsdl18)

**엔트로피는 이 정보량의 평균**

Entropy = - p(a)\*log(p(a)) - p(b)\*log(p(b))

0.99 \* -np.log(0.99) + 0.01 \* -np.log(0.01)

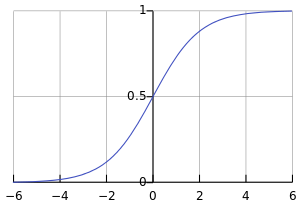
1. 근데 threshold 값들을 모두 계산하려면 너무 비용이 많이듬.

* random하게 forest를 만듬(이를 ensemble이라 함)

1. 이를 logestic function (x가 0일 때 항상 값이 반. 0.5 그리고 갈수록 0 과 1로 수렴함)에 넣어 분류

Logestic regression – 선형회귀와 다르게 결과가 범주형일 때.

그래프 생긴거 보듯이, 만약 data가 치우쳐져 있을 때, 선형보다는 저렇게 하는 게 더 정확해서



3)Clustering : 분류처럼 모든 데이터 값은 어떤 특징의 값을 가지고 있음. Label이 없음.

Cluster는 비슷한 특징을 가진 데이터의 집단

knn처럼 거리로 계산함.

clustering에는 많은 방법들이 있는데 그 중에 k-mean을 배움

k개의 중간점에서 각 점들의 거리(유클리드 이용)를 error라고 부름. 그거의 제곱을 squared error. 그걸 합친 걸 SSE(sum of squared error). 사실상 **sse를 구하는 거는 불가능(모든 가능성을 다 계산하지 않고서는 불가능한 지 찾을 수 없음.(nonlinear optimization problem NLP) 데이터 양이 많으면 비용이 너무 많이 듬).**

**왜? 불가능?** k점들을 어디서 잡는지에 따라 결과가 매우 달라짐. -> 예측할 수가 없이 다 계산 해 보아야함

알고리즘 – 1. K개 랜덤하게 뽑아서, 그 k개에서 어디에 제일 가까운지를 정해서 소속을 정함.

2. 각 clusters의 k점부터의 거리의 평균을 구해서 그 중간을 centroid로 정함.

3. 새로이 선정된 k로 다시 점들을 분류함.

4. 데이터들이 수렴할 때(더 이상 변화가 없을 때) 알고리즘 계산이 멈추고 그걸 최종으로 함. SSE

-> SSE를 찾는 것이 불가능 하니까 (non linear해서) 그걸 근사함. 그리고 k의 개수를 어떻게 고르는 지 생각해 봄.

Anomany detection – 억지로 끼워넣은 듯한 점을 부름

Clustering 응용 – 그림 압축할 때 색을 k개로 묶어서. (Vector Quantization이라고 부름. RGB 3차원이라 vector)

4)LDA : Linear Discriminant Analysis

**Whitening transformation** - 점들을 무수한 선에 직각으로 연결해서 가장 분리가 쉬운 점을 찾아냄-> 2차원 데이터를 1차원으로

3페이지에 - > x의 차이가 y의 차이보다 크지만 x만에 중복되는 겹치는 데이터가 y에서 겹치는 데이터보다 많음. => x,y만 가지고 부족하다 =>> within class scatter

그래서 나온게 **SVM(Support Vector Machines)**

데이터 들을 분류할 때 그 거리들의 중간의 optimal한 선을 찾음 근데 나중의 정확성들을 위해서 margin을 최대화 하는 것을 찾아야 댐