**3. 모델 설명**

**A. Logistic Regression**

로지스틱 회귀분석(logistic regression)은 종속변수가 명목변수일 때 사용하는 회귀분석 방법입니다. 회귀분석과 모든 형태가 같고 단지 종속변수만 이항형 또는 순서적인 다항형인 경우에 사용합니다. 로지스틱 회귀분석은 회귀분석이라는 명칭과 달리 회귀분석 문제와 분류문제 모두에 사용할 수 있습니다.

로지스틱 함수는 음의 무한대부터 양의 무한대까지의 실수값을 0부터 1사이의 실수값으로 1 대 1 대응시키는 시그모이드 함수입니다. 보통 시그모이드 함수라고 하면 로지스틱 함수를 가리킵니다. 베르누이 시도에서 1이 나올 확률 μ와 0이 나올 확률 1−μ의 비율(ratio)을 승산비, 오즈비(odds ratio)라고 합니다. (odds ratio=μ/(1−μ)) 0부터 1사이의 값만 가지는 μ를 승산비로 변환하면 0부터 양의 무한대까지의 값을 가질 수 있습니다. 승산비를 로그 변환한 것이 로짓함수(Logit function)다. 로짓함수의 값은 로그 변환에 의해 음의 무한대(−∞)부터 양의 무한대(∞)까지의 값을 가질 수 있습니다.

로지스틱 함수(Logistic function)는 로짓함수의 역함수입니다. 즉, 음의 무한대(−∞)부터 양의 무한대(∞)까지의 값을 가지는 입력변수를 0부터 1사의 값을 가지는 출력변수로 변환하는 것입니다.

**B. SGD-Classifier(Stochastic Gradient Descent - Classifier)**

SGD-Classifier는 SGD에 의해 최적화된 선형 분류기(SVM, 로지스틱 회귀)입니다. 이것은 두 가지 다른 개념입니다. SGD가 최적화 방법인 반면 로지스틱 회귀 또는 선형 지원 벡터 머신은 기계 학습 알고리즘/모델입니다. 머신 러닝 모델은 손실 함수를 정의하고 최적화 방법은 이를 최소화/최대화한다고 생각할 수 있습니다.

경사 하강법은 비용 함수를 최소화하는 데 사용됩니다. 경사 하강법은 최적화를 수행하는데 사용되는 알고리즘 중 하나이며 신경망을 최적화하는 가장 일반적인 방법입니다. 그러나 이러한 종류의 알고리즘을 사용하여 Logistic Regression 및 Support Vecotor Machines와 같은 선형 분류기를 최적화할 수도 있습니다.

경사 하강법에는 세 가지 유형 Batch gradient descent, Stochastic gradient descent, Mini-batch gradient descent가 있습니다. 여기서 Loss Function을 계산할 때 전제 Train-Set을 사용하는 것을 Batch Gradient Descent라고 한다. 그러나 이렇게 계산하면 한번 step을 내딛을 때, 전체 데이터에 대해 Loss Function을 계산해야 하므로 너무 많은 계산양을 필요로 한다. 이를 방지하기 위해 보통은 Stochastic Gradient Descent(SGD) 방법을 사용한다. 이 방법에서는 Loss Function을 계산할 때, 전체 데이터(Batch) 대신 일부 데이터의 모음(Mini-Batch)를 사용하여 Loss Function을 계산한다. Batch Gradient Descent보다 다소 부정확할 수는 있지만, 계산 속도가 훨씬 빠르기 때문에 같은 시간에 더 많은 step을 갈 수 있으며, 여러 번 반복할 경우 Batch 처리한 결과로 수렴한다. 또한 Batch Gradient Descent에서 빠질 Local Minima에 빠지지 않고 더 좋은 방향으로 수렴할 가능성도 높다.

Logistic Regression의 비용 함수의 최소값은 직접 계산할 수 없으므로 Stochastic Gradient Descent를 통해 최소화하려고 합니다. 이 과정에서 각 훈련 관찰에 대해 비용 함수를 따라 최소값으로 내려갑니다.

**C. KNN(K-Nearest Neighbor)**

K-Nearest Neighbor(KNN)은 지도 학습 알고리즘 중 하나입니다. 어떤 데이터가 주어지면 그 주변(이웃)의 데이터를 살펴본 뒤 더 많은 데이터가 포함되어 있는 범주로 분류하는 방식입니다. 새로운 데이터가 주어졌을 때 이를 어떠한 Class로 분류할지 판단하는 문제입니다. KNN은 K의 결정방법에 따라 결과 값이 바뀔 수 있습니다. K가 너무 작아서도 안 되고, 너무 커서도 안 됩니다. K의 default 값은 5입니다. 가장 가까운 주변 5개 데이터를 기반으로 분류한다는 것입니다. 일반적으로 K는 홀수를 사용합니다. 짝수일 경우 동점이 되어 하나의 결과를 도출할 수 없기 때문입니다. KNN에서는 데이터와 데이터 사이의 거리를 구해야 합니다. 거리를 구하는 방식은 두 가지, 유클리드 거리(Euclidean Distance), 맨해튼 거리(Manhattan Distance)가 있습니다.

**D. SVM(Support Vector Machine)**

Support Vector Machine은 결정 경계(Decision Boundary), 즉 분류를 위한 기준 선을 정의하는 모델이다. 그래서 분류되지 않은 새로운 점이 나타나면 경계의 어느 쪽에 속하는지 확인해서 분류 과제를 수행할 수 있게 됩니다. 만약 데이터에 2개 속성(feature)만 있다면 결정 경계는 간단한 선 형태가 될 것이고, 속성이 3개로 늘어난다면 이 때의 결정 경계는 ‘선’이 아닌 ‘평면’이 된다. 차원, 즉 속성의 개수가 늘어날수록 결정 경계도 단순한 평면이 아닌 고차원이 될 텐데 이를 “초평면(hyperplane)”이라고 부른다.

Support Vectors는 결정 경계와 가까이 있는 데이터 포인트들을 의미하고, 마진(Margin)은 결정 경계와 서포트 벡터 사이의 거리를 의미한다. 여기서 최적의 결정 경계는 ​가중치 벡터 w와 직교하면서 margin이 최대가 되는 선형을 찾는다.

**E. Decision Tree**

결정 트리(Decision Tree, 의사결정트리, 의사결정나무라고도 함)는 분류(Classification)와 회귀(Regression) 모두 가능한 지도 학습 모델 중 하나입니다. 결정 트리는 스무고개 하듯이 예/아니오 질문을 이어가며 학습합니다. 이렇게 특정 기준(질문)에 따라 데이터를 구분하는 모델을 결정 트리 모델이라고 합니다. 한번의 분기 때마다 변수 영역을 두 개로 구분합니다. 결정 트리에서 질문이나 정답을 담은 네모 상자를 노드(Node)라고 합니다. 맨 처음 분류 기준 (즉, 첫 질문)을 Root Node라고 하고, 맨 마지막 노드를 Terminal Node 혹은 Leaf Node라고 합니다.

엔트로피(Entropy)는 불순도(Impurity)를 수치적으로 나타낸 척도입니다. 분기 이전의 엔트로피에서 분기 이후의 엔트로피를 뺀 수치를 정보 획득이라고 합니다. 결정 트리 알고리즘은 정보 획득을 최대화하는 방향으로 학습이 진행됩니다.

따라서 첫 분기점을 정보 획득을 최대화하는 기준으로 잡습니다. 이런식으로 max\_depth나 min\_sample\_split으로 설정한 범위까지 분기를 하게 됩니다. 이것이 바로 결정트리의 전체적인 알고리즘입니다.

**F. Pruning**

오버피팅을 막기 위한 전략으로 가지치기(Pruning)라는 기법이 있습니다. 트리에 가지가 너무 많다면 오버피팅이라 볼 수 있습니다.

사전 가지치기: 최대 깊이나 터미널 노드의 최대 개수, 혹은 한 노드가 분할하기 위한 최소 데이터 수를 제한하는 것입니다. min\_sample\_split = 10이면 한 노드에 10개의 데이터가 있다면 그 노드는 더 이상 분기를 하지 않습니다. 또한, max\_depth를 통해서 최대 깊이를 지정해줄 수도 있습니다. max\_depth = 4이면, 깊이가 4보다 크게 가지를 치지 않습니다.

사후 가지치기: 결과를 본 후 노드를 지우거나 깊이를 조정하는 것. 가지치기는 사전 가지치기와 사후 가지치기가 있지만 sklearn에서는 사전 가지치기만 지원합니다.

**G. Random Forest**

랜덤 포레스트의 포레스트는 숲(Forest), 결정 트리는 나무(Tree)입니다. 나무가 모여 숲을 이룹니다. 즉, 결정 트리(Decision Tree)가 모여 랜덤 포레스트(Random Forest)를 구성합니다. 결정 트리 하나만으로도 머신러닝을 할 수 있습니다. 하지만 결정 트리의 단점은 훈련 데이터에 오버피팅이 되는 경향이 있다는 것입니다. 여러 개의 결정 트리를 통해 랜덤 포레스트를 만들면 오버피팅 되는 단점을 해결할 수 있습니다.

예를 들어 Feature가 30개라고 합시다. 30개의 Feature를 기반으로 하나의 결정 트리를 만든다면 트리의 가지가 많아질 것이고, 이는 오버피팅의 결과를 야기할 것입니다. 하지만 30개의 Feature 중 랜덤으로 5개의 Feature만 선택해서 하나의 결정 트리를 만들고, 또 30개 중 랜덤으로 5개의 Feature를 선택해서 또 다른 결정 트리를 만들고... 이렇게 계속 반복하여 여러 개의 결정 트리를 만들 수 있습니다. 결정 트리 하나마다 예측 값을 내놓겠죠. 여러 결정 트리들이 내린 예측 값들 중 가장 많이 나온 값을 최종 예측값으로 정합니다. 다수결의 원칙에 따르는 것입니다. 이렇게 의견을 통합하거나 여러 가지 결과를 합치는 방식을 앙상블(Ensemble)이라고 합니다. 즉, 하나의 거대한 (깊이가 깊은) 결정 트리를 만드는 것이 아니라 여러 개의 작은 결정 트리를 만드는 것입니다. 여러 개의 작은 결정 트리가 예측한 값들 중 가장 많은 값(분류일 경우) 혹은 평균값(회귀일 경우)을 최종 예측 값으로 정하는 것입니다.