ML Day5, 6 (수식 포함 용어 정리)

1. **Underfitting** : 너무 단순한 모델
2. **Overfitting**: 너무 복잡한 모델

- **Mean subtraction(평균차감)**

  데이터의 모든 feature 각각에 대해서 평균 값 만큼 차감하는 방법, 기하학 관점에서 보자면 데이터 군집을 모든 차원에 대해서 원점으로 이동시키는 것으로 해석할 수 있음 [X -= np.mean(X, axis = 0)]

- **Square_of_mean** = mean**2
- **Mean_of_square** = (a**2 + b**2 + c**2) / len(a, b, c)
- **Variance** = mean_of_square - square_of_mean
- **Score_std** = score_variance**0.5 (표준편차: 분산에 루트를 씌운 값)

▼ **Mean of Square, Square of mean (제곱의 평균, 평균의 제곱)**

```
score1 = 10
score2 = 20
score3 = 30
n_student = 3
score_mean = (score1 + score2 + score3) / n_student
square_of_mean = score_mean**2
mean_of_square = (score1**2 + score2**2 + score3**2)/n_student
score_variance = mean_of_square - square_of_mean
score_std = score_variance**0.5
print("mean:", score_mean)
print("variance:", score_variance)
print("standard deviation:", score_std)
```

▼ score_mean : 각 score의 값을 더해준 후 n_student로 나눠준 값

▼ square_of_mean : 평균의 제곱

▼ mean_of_square : 제곱의 평균

▼ score_variance : 제곱의 평균 - 평균의 제곱 (분산)

▼ score_std : 분산**0.5 (표준편차)

```
mean: 20.0
variance: 66.66666666666669
standard deviation: 8.16496580927726
```

▼ **Standardization(표준화)** - normalization(정규화)과 같이 머신러닝 알고리즘을 훈련시키는데 있어서 사용되는 특성들, 비슷한 영향력을 행사하도록 값을 변환해주는 기술 —> 특성 스케일링(feature scaling), 데이터 스케일링(data scaling)

```
score1 = 10
score2 = 20
score3 = 30
n_student = 3
score_mean = (score1 + score2 + score3)/n_student
square_of_mean = score_mean**2
mean_of_square = (score1**2 + score2**2 + score3**2)/n_student
score_variance = mean_of_square - square_of_mean
score_std = score_variance**0.5
print("mean:", score_mean)
print("standard deviation:", score_std)

score1 = (score1 - score_mean)/score_std     #1
score2 = (score2 - score_mean)/score_std
score3 = (score3 - score_mean)/score_std
```

```
score_mean = (score1 + score2 + score3)/n_student   #2
square_of_mean = score_mean**2
mean_of_square = (score1**2 + score2**2 + score3**2)/n_student
score_variance = mean_of_square - square_of_mean
score_std = score_variance**0.5
print("mean:", score_mean)
print("standard deviation:", score_std)
```

▼ #1 : 각 score값에서 평균을 뺀값을 표준편차로 나눠준후 다시 score값에 대입

▼ #2 : 위 code의 결과로 바뀐 score값들로 다시 평균, 분산, 표준편차를 구해줌 (standardization)

```
mean: 20.0
standard deviation: 8.16496580927726
mean: 0.0
standard deviation: 1.0
```

▼ Vector - Vector Operations

```
x1, y1, z1 = 1, 2, 3
x2, y2, z2 = 3, 4, 5
x3, y3, z3 = x1 + x2, y1 + y2, z1 + z2
x4, y4, z4 = x1 - x2, y1 - y2, z1 - z2
x5, y5, z5 = x1 * x2, y1 * y2, z1 * z2
print(x3, y3, z3)
print(x4, y4, z4)
print(x5, y5, z5)
```

```
4 6 8
-2 -2 -2
3 8 15
```

▼ Vector norm

```
x, y, z = 1, 2, 3
norm = (x**2 + y**2 + z**2)**0.5
print(norm)
```

```
3.7416573867739413
```

▼ Making Unit Vectors

```
x, y, z = 1, 2, 3
norm = (x**2 + y**2 + z**2)**0.5
print(norm)

x, y, z = x/norm, y/norm, z/norm
norm = (x**2 + y**2 + z**2)**0.5
print(norm)
```

```
3.7416573867739413
1.0
```

▼ Dot Product (Vector 또는 행렬들 간의 내적)

```
x1, y1, z1 = 1, 2, 3
x2, y2, z2 = 3, 4, 5

dot_prod = x1*x2 + y1*y2 + z1*z2
print(dot_prod)
```

```
26
```

▼ Euclidean Distance

```
x1, y1, z1 = 1, 2 ,3
x2, y2, z2 = 3, 4, 5

e_distance = (x1 - x2)**2 + (y1 - y2)**2 + (z1 - z2)**2
e_distance **= 0.5

print(e_distance)
```

```
3.4641016151377544
```

▼ Squared Error (예측값 - 정답)의 제곱

```
pred1, pred2, pred3 = 10, 20, 30
y1, y2, y3 = 10, 25, 40

s_error1 = (pred1 - y1)**2
s_error2 = (pred2 - y2)**2
s_error3 = (pred3 - y3)**2

print(s_error1, s_error2, s_error3)
```

```
0 25 100
```