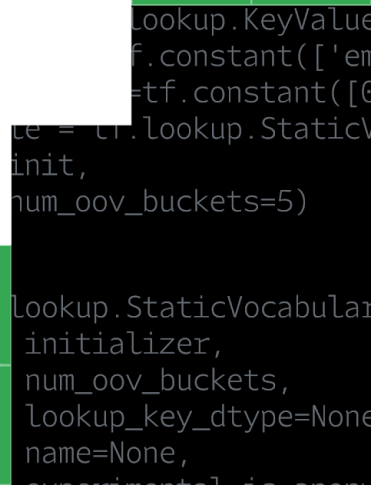
Google Developer Student Clubs

1주차 2조

팀원: 황동욱, 권도혁



(황동욱) 알고리즘 정리

🤔Random Forest는 왜 손실함수가 없지?

랜덤 포레스트는 결정 트리의 앙상블 기법으로, 기본적으로 각 트리가 데이터를 분할하는 데 있어서 순도 증가나 정보 획득을 최대화하는 방향으로 학습합니다. 이때의 "순도 증가"나 "정보 획득"은 분류 작업의 경우에 사용되는 기준입니다. 반면 회귀 작업의 경우 분산의 감소를 목표로 합니다.

이렇게 보면, 랜덤 포레스트는 간접적으로 손실을 최소화하는 방향으로 학습되긴 하지만, 그래디언트 부스팅처럼 명시적인 수학적 손실 함수를 기반으로 한 순차적인 학습 방식을 사용하지는 않습니다.

TensorFlow나 다른 라이브러리와 관련 없이, 랜덤 포레스트의 기본 알고리즘 자체는 명시적인 손실 함수를 사용하지 않습니다. 이와 대조적으로 그래디언트 부스팅은 명시적인 손실 함수에 대한 잔차를 줄이는 방향으로 순차적으로 트리를 학습시킵니다.

🤔tfdf를 쓰게 되면 feature scaling은 필요 없나?

예, 맞습니다. TensorFlow Decision Forests (TFDF)와 같은 트리 기반의 알고리즘을 사용할 때는 특성 스케일링이 일반적으로 필요하지 않습니다. 트리 알고리즘은 각 특성의 임계값을 기준으로 데이터를 분할하기 때문에 스케일의 영향을 받지 않습니다.

전통적인 선형 모델, 신경망, SVM(Support Vector Machine) 등의 알고리즘을 사용할 때는 특성 스케일링이 중요합니다. 이러한 알고리즘들은 특성의 스케일에 민감하게 반응하기 때문에, 스케일링을 통해 모델의 학습 속도와 성능을 향상시킬 수 있습니다.

그러나, TFDF나 다른 트리 기반의 알고리즘을 사용할 때는 이러한 스케일링 과정을 건너뛸 수 있습니다.

🤔그럼, 다른 모델들은? feature scaling 해야 하는지...

1. **TensorFlow (Deep Learning)**:
   - **필요함:** 딥러닝 모델, 특히 신경망은 입력 특성의 스케일에 매우 민감합니다. 스케일링을 하지 않으면 모델 학습이 불안정할 수 있고, 수렴하는 데 더 많은 시간이 걸릴 수 있습니다.
2. **TensorFlow Decision Forests (TFDF)**:
   - **필요 없음**: TFDF는 트리 기반 알고리즘을 사용하므로 특성 스케일링이 필요하지 않습니다. 트리는 특성의 임계값을 기준으로 데이터를 분할하기 때문에 스케일의 영향을 받지 않습니다.
3. **Scikit-learn**:
   - **필요함** (일부 모델에 대해):
     - **선형 모델, 로지스틱 회귀, SVM, k-최근접 이웃 (KNN), 신경망** 등은 특성 스케일링에 민감합니다.
   - **필요 없음** (일부 모델에 대해):
     - **결정 트리, 랜덤 포레스트, 그래디언트 부스팅 트리**와 같은 트리 기반의 알고리즘은 특성 스케일링을 필요로 하지 않습니다.
4. **PyTorch (Deep Learning)**:
   - **필요함**: PyTorch를 사용하여 딥러닝 모델을 구현할 때는 TensorFlow와 마찬가지로 특성 스케일링이 필요합니다.

🤔 그럼, scaling을 해서 나쁠건 없다?

특성 스케일링을 하는 것이 대부분의 경우에 문제를 일으키지 않습니다. 실제로, 스케일링을 하면 많은 알고리즘들의 성능이 향상되거나 학습 속도가 빨라집니다. 그러나 아래와 같은 몇 가지 주의 사항을 알고 있어야 합니다:

1. **정보 손실**: 원시 데이터의 절대적인 크기나 범위가 중요한 정보를 포함하고 있는 경우, 스케일링을 통해 그 정보를 잃어버릴 수 있습니다.
2. **해석 용이성**: 스케일링된 데이터는 원래의 단위와는 다르므로, 결과를 해석할 때 추가적인 변환 단계를 고려해야 할 수 있습니다.
3. **트리 기반 모델**: 트리 기반 알고리즘은 특성 스케일링의 영향을 받지 않습니다. 그러나 스케일링을 해도 모델의 성능에는 큰 영향을 미치지 않습니다.
4. **스케일링 방법**: 다양한 스케일링 방법들 (예: Min-Max 스케일링, Z-score 정규화, 로그 변환 등)이 있으며, 각 방법이 데이터에 미치는 영향이 다를 수 있습니다. 따라서 가장 적합한 스케일링 방법을 선택하는 것이 중요합니다.

결론적으로, 특성 스케일링은 대부분의 경우에 안전하며, 성능 향상을 위해 권장되는 전처리 단계입니다. 그러나 사용하는 모델과 문제의 특성에 따라 적절한 스케일링 방법을 선택해야 합니다.

🤔 skewness를 조정하는 건 어떤 모델에 대해서 필요하지?

Skewness(왜도)는 데이터셋의 분포가 얼마나 치우쳐져 있는지를 나타내는 척도입니다. 왜도는 다음과 같은 모델들에 영향을 줄 수 있습니다:

1. **선형 회귀** 및 **로지스틱 회귀**: 이러한 모델은 가정 중 하나로 종속 변수와 독립 변수 사이의 선형 관계가 있다는 것입니다. 치우친 특성은 이러한 관계를 왜곡시킬 수 있으므로, 왜도를 보정하는 것이 종종 유용합니다.
2. **신경망 (Neural Networks)**: 특성의 스케일과 분포는 신경망의 학습에 큰 영향을 줄 수 있습니다. 치우친 특성은 학습 속도를 늦출 수 있으므로 로그 변환 같은 방법을 사용하여 왜도를 줄이는 것이 좋습니다.
3. **나이브 베이즈 분류기**: 이 분류기는 특성의 분포에 대한 가정을 합니다. 치우친 특성은 이러한 가정을 위반할 수 있으므로 왜도 보정이 필요할 수 있습니다.
4. **트리 기반 알고리즘 (예: 결정 트리, 랜덤 포레스트, 그래디언트 부스팅)**: 트리 기반 모델은 특성의 분포에 대한 가정을 하지 않으므로, 왜도에 크게 민감하지 않습니다. 따라서 왜도 보정은 이러한 모델에 대해서는 그리 중요하지 않을 수 있습니다.
5. **SVM (Support Vector Machines)**: SVM은 특성의 스케일에 민감하기 때문에, 치우친 특성은 모델의 성능에 영향을 줄 수 있습니다. 왜도를 보정하는 것이 좋을 수 있습니다.
6. **K-NN (K-Nearest Neighbors)**: K-NN은 거리 기반 알고리즘이므로, 특성의 스케일과 분포에 매우 민감합니다. 왜도를 보정하는 것이 유용할 수 있습니다.

결론적으로, 왜도는 대부분의 모델에 영향을 줄 수 있지만, 트리 기반 알고리즘은 그렇게 민감하지 않을 수 있습니다. 왜도가 높은 특성이 있을 경우, 해당 특성을 변환하여 왜도를 줄이는 것을 고려해 보는 것이 좋습니다.

🤔 skewness를 조정해서 나쁠건 없다?

특성의 왜도를 보정하는 것은 대부분의 경우 모델의 성능 향상에 도움을 줄 수 있지만, 항상 그런 것은 아닙니다. 왜도 보정의 장단점을 살펴보겠습니다:

장점:

1. **모델 성능 향상**: 치우친 데이터를 보정함으로써, 특히 선형 모델이나 거리 기반 알고리즘과 같은 모델의 성능이 향상될 수 있습니다.
2. **수렴 속도**: 신경망과 같은 반복적인 학습 알고리즘에서, 보정된 데이터는 보통 더 빠른 수렴 속도를 가져올 수 있습니다.
3. **모델 해석력**: 보정된 특성은 종종 원래의 치우친 특성보다 더 해석하기 쉬울 수 있습니다.

단점:

1. **데이터 왜곡**: 로그 변환과 같은 변환은 데이터의 원래 의미나 분포를 왜곡시킬 수 있습니다. 이로 인해 결과의 해석이 더 어려워질 수 있습니다.
2. **성능 저하**: 트리 기반 알고리즘과 같은 몇몇 모델은 치우친 데이터에서도 잘 작동할 수 있습니다. 이런 경우, 왜도를 보정하는 것이 성능을 오히려 저하시킬 수도 있습니다.
3. **과정 복잡도**: 왜도 보정은 데이터 전처리 단계에 추가적인 복잡성을 가져올 수 있습니다.

결론적으로, 왜도 보정은 많은 경우에 도움이 될 수 있지만, 항상 그렇다고는 할 수 없습니다. 따라서 왜도 보정을 적용하기 전과 후의 모델 성능을 비교해 보는 것이 중요합니다.

🤔 상관계수도 종류가 있었네!

1. **피어슨 상관계수 (Pearson correlation coefficient)**:
   - 선형 관계를 측정합니다. 두 변수 간의 선형적인 연관성이 얼마나 강한지 알려줍니다.
   - 값은 -1에서 1 사이입니다. 1은 완벽한 양의 선형 관계를, -1은 완벽한 음의 선형 관계를 의미하며, 0은 선형 관계가 없음을 나타냅니다.
   - 원래의 데이터 값을 기반으로 계산됩니다.
   - 아웃라이어에 민감하다는 단점이 있습니다.
2. **스피어만 순위 상관계수 (Spearman rank correlation coefficient)**:
   - 변수들의 순위에 기반한 비선형 관계를 측정합니다.
   - 값은 -1에서 1 사이입니다. 이는 피어슨 상관계수와 마찬가지로 관계의 방향과 강도를 나타냅니다.
   - 원래의 데이터 값 대신 순위를 기반으로 계산됩니다.
   - 아웃라이어에 피어슨 상관계수보다는 덜 민감하며, 비선형 관계에 더 적합합니다.

다른 상관계수로는 "켄달의 타우 (Kendall's Tau)"도 있습니다. 스피어만 상관계수와 비슷하게 순위를 기반으로 하지만, 연산 방식이 조금 다릅니다.

어떤 상관계수를 사용할지 결정할 때는 데이터의 특성과 관계의 형태를 고려해야 합니다. 선형 관계가 의심될 때는 피어슨 상관계수를, 비선형 관계나 순위 데이터를 사용할 때는 스피어만 또는 켄달의 타우를 사용하는 것이 일반적입니다.

(권도혁) Data 분석 정리

데이터 구성

문제 배경

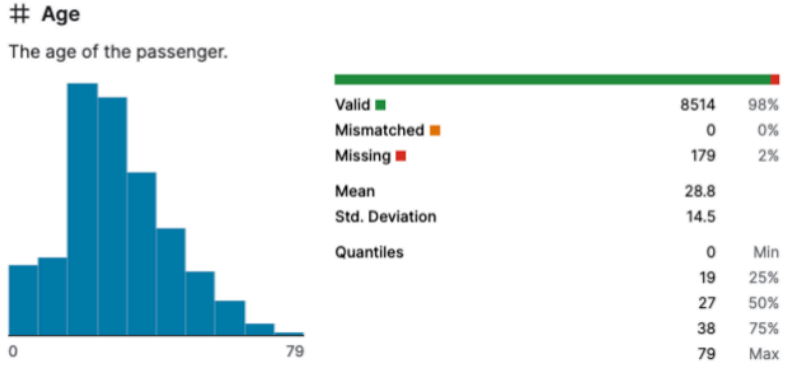
우주선 타이타닉 → 날라가다가 사고나서 승객 절반이 다른 차원으로 날라감

다양한 정보를 통해 각 승객이 날라갔는지 안갔는지 True False 예측

(test data 4277명)

Data

- train.csv (8693명)
  - PassengerID [gggg_pp] : 고유 ID
    - 우선 training, 분석 에 포함하지 않는 것이 좋아보임.
    - 동일 group 이면 같은 결과가 나타나는 경향이 있을경우 개선 요소로 포함할 수는 있을 것 같음
  - HomePlanet [Earth, Europa, Mars, (Null)] : 출발 행성
    - Earth 53%, Europa 25%, Others 23%
  - CryoSleep [true, false, (null)] : 냉동 수면 여부
    - T 35%, F 63%, Null 2%
  - Cabin [deck/num/side] :
    - Data Preprocessing 필요
    - null 2% → Missing Value 처리 어려움 (후순위)
    - Deck : alphabet (A~G)
    - num : Integer (0~600)
    - side : P(Port) or S(Starboard)
  - Destination [TRAPPIST-1e, 55 Cancri e, PSO J318.5-22]
    - T 68%, 55 21%, Others 11%
  - Age



    - 0 ~ 79
    - Feature 분석시 우선고려
  - VIP [true, false, null]
    - true 2%, false 95%, null 2%
  - `RoomService` , `FoodCourt` , `ShoppingMall` , `Spa` , `VRDeck`
    - 사용금액 → 정렬로 representation 얻거나 특징 없으면 우선 학습 제외
  - Name
    - 우선 training 제외

Missing value를 처리하기 위해서 어떤 방법을 이용해야 할까

1. Data imputation

: Missing Value(결측치) 처리 방법

[참고] https://www.theanalysisfactor.com/seven-ways-to-make-up-data-common-methods-to-imputing-missing-data/

1. Mean

   : 평균값을 이용

2. Substitution

   : 샘플링 되지 않은 데이터에서 가져옴 → 우리 경우에는 적용 X

3. **Hot deck**

   : 다른 변수에서 비슷한 값을 갖는 데이터 중에서 하나를 랜덤 샘플링 하여 그 값을 복사해오는 방법.

   → 결측치가 존재하는 변수가 가질 수 있는 값의 범위가 한정되어 있을 때 이용하면 좋음.

4. Cold deck

   : 다른 변수에서 비슷한 값을 갖는 데이터에서 값을 가져오는데 랜덤하지 않고 일정한 규칙에 따라 가져옴. (K번째 등등)

5. Regression

   : 결측치가 존재하지 않는 변수를 feature, 채우고자 하는 변수를 target으로 regression

   → 결측치가 존재하는 변수가 임의의 값을 가지는 경우에 이용하면 좋을거 같음.

6. Stochastic regression

   : Regression으로 찾은 value에 randomness를 추가하기 위해 residual value를 더해서 이용.

7. Interpolation and extrapolation

   : 같은 대상으로부터 얻은 관측치로부터 결측치 추정 ex)한 아이의 성장과정 데이터 등에서 키와 몸무게 상관관계 이용 등

Hot deck 방법을 이용

우선 Missing value가 존재하는 Feature 정리 (Name은 제외하고)

Hot deck 가능	HomePlanet, CryoSleep, Destination, VIP, RoomService, FoodCourt, ShoppingMall, Spa, VRDeck, Deck, Side
Hot deck 어려움	Age, Cabin_num

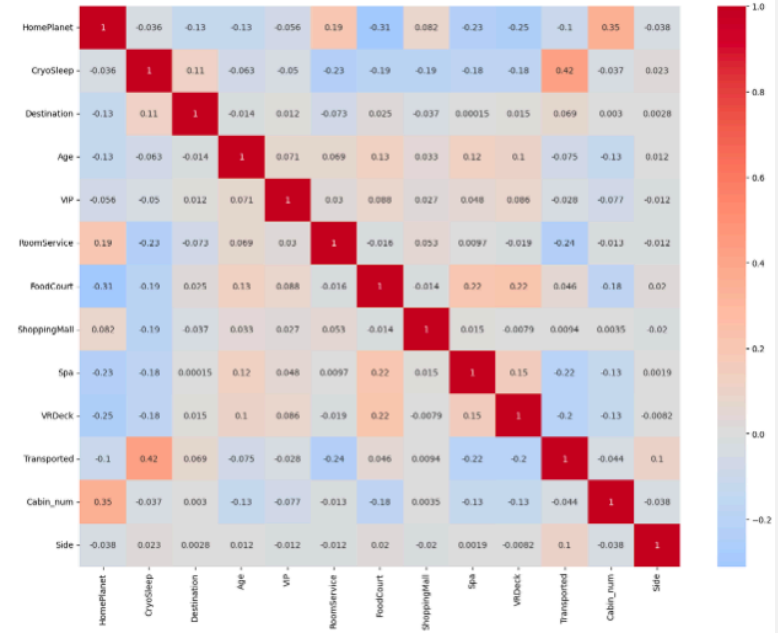
Missing value를 처리하기 위해서 어떤 방법을 이용해야 할까

어떤 feature를 기준으로 채울 것인지?

1. Hot deck imputation이 의미가 있으려면 많은 feature에서 동일한 값, 또는 유사한 값을 갖는 data를 골라서 imputation을 해야함. (예. CryoSleep, Destination, VIP 이 세 값이 모두 같은 사람의 Homeplanet 정보를 가져온다.) 그러나, 어떤 값을 기준으로 할지 애매하고, 얼마나 많은 feature들에서 유사한 값을 갖는 데이터를 찾아낼 것인가가 결정하기 어려운 부분이다.
2. 우선적으로 Transported 결과에 가장 영향을 많이주는 feature를 찾아 해당 값을 중심으로 생각하는 것이 유리함. 피어슨, 스피어만 상관계수를 Heat map으로 출력하여 'CryoSleep'이 매우 관계가 높으며, 'Destination'과 'Side'역시 의미있는 결과를 보임. ('CryoSleep'의 경우 냉동 수면 상태이면 Transported 되지 않았다는 나름 의미있는 정보를 담고있음)

Missing value를 처리하기 위해서 어떤 방법을 이용해야 할까

[피어슨 상관계수 히트맵]



[스피어만 상관계수 히트맵]