

2주차 2조

팀원: 황동욱, 권도혁

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([0  
ce = tf.lookup.StaticV  
init,  
num_oov_buckets=5)
```

```
lookup.StaticVocabular  
initializer,  
num_oov_buckets,  
lookup_key_dtype=None  
name=None,  
experimental_is_open
```

(황동욱) 알고리즘 정리



randomforest 모델을 쓸때 생각보다 feature를 늘려도 overfitting이 안되네 이유가 뭐야?

RandomForest 모델은 여러 개의 의사결정 나무(decision trees)를 통합하여 예측을 수행하는 앙상블 방법입니다. **RandomForest** 모델이 과적합(overfitting)에 상대적으로 덜 민감한 주요 이유들은 다음과 같습니다:

1. **부트스트랩 샘플링 (Bootstrap Sampling)**: 랜덤 포레스트는 각 트리를 훈련할 때 원래 데이터셋에서 부트스트랩 샘플(중복을 허용한 랜덤 샘플)을 추출하여 사용합니다. 따라서 각 트리는 약간씩 다른 데이터를 기반으로 학습되며, 이로 인해 **트리 간의 다양성이 증가하고 과적합이 감소**합니다.
2. **특성의 무작위 선택**: 랜덤 포레스트의 각 트리는 분할 포인트를 결정할 때 전체 특성 중 일부만을 무작위로 선택하여 고려합니다. 이로 인해 트리가 특정 특성에 지나치게 의존하는 것을 방지하고, 다양성을 증가시켜 과적합 위험을 줄입니다.
3. **트리의 수**: 랜덤 포레스트는 많은 트리를 사용하여 예측을 집계(평균)합니다. 여러 트리의 예측을 결합하면 트리 개별의 **노이즈나 변동이 평균화되므로, 과적합을 줄이는 효과**가 있습니다.
4. **각 트리의 깊이**: 랜덤 포레스트에서는 트리를 완전히 성장시키는 것이 일반적입니다(즉, 가지치기를 수행하지 않습니다). 이렇게 하면 **각 트리는 개별적으로는 과적합될 수 있지만**, 전체 포레스트는 위에 언급한 **다양성 때문에 과적합에 덜 민감**하게 됩니다.
5. **OOB (Out-Of-Bag) 오류**: 랜덤 포레스트는 부트스트랩 샘플링으로 인해 훈련 중에도 테스트 세트와 유사한 **OOB 샘플을 사용하여 모델의 성능을 평가할 수 있습니다**. 이를 통해 모델의 과적합 상태를 감지하고 조정할 수 있습니다.

🤔 randomforest를 쓸 때 내가 수치형 데이터를 범주를 나누어서 categorical로 바꾼다고 해서 성능이 좋아질 가능성이 존재해?

RandomForest 와 같은 트리 기반 모델에서 수치형 데이터를 범주형(categorical)으로 변환하는 것이 모델 성능에 긍정적인 영향을 미칠 수 있는 경우가 있습니다. 그러나 이러한 변환이 성능 향상을 보장하는 것은 아니며, 데이터와 문제의 특성에 따라 다르게 작용할 수 있습니다. 이러한 변환을 고려하는 이유와 주의 사항은 다음과 같습니다:

1. 비선형 관계: 특성의 수치가 비선형적인 관계를 가질 때, 이를 여러 범주로 나누어 표현하면 모델이 이 관계를 더 잘 포착할 수 있습니다. 예를 들어, 연령을 여러 연령대로 나누면 특정 연령대의 특별한 패턴을 더 쉽게 인식할 수 있습니다.
2. 모델의 복잡도: 수치형 데이터를 범주형 데이터로 변환하면, 특정 경우에는 모델의 복잡도가 줄어들어 과적합(overfitting) 위험이 감소할 수 있습니다.
3. 의미 있는 구간: 도메인 지식을 활용하여 수치형 데이터를 의미 있는 범주로 나눌 수 있다면, 모델이 이 정보를 더 효과적으로 활용할 수 있습니다.

주의 사항:

1. 정보의 손실: 수치형 데이터를 범주형으로 변환하면 원래의 수치 정보 중 일부가 손실될 수 있습니다. 이로 인해 모델의 성능이 하락할 수도 있습니다.
2. 적절한 구간 선택: 너무 많은 범주로 나누거나 너무 큰 범주로 나누면, 모델 성능에 부정적인 영향을 미칠 수 있습니다. 도메인 지식이나 데이터 탐색을 통해 적절한 구간을 선택하는 것이 중요합니다.
3. 카디널리티: 변환 후의 범주의 수가 많아지면, 모델의 복잡도와 훈련 시간이 증가할 수 있습니다.

최종적으로, 수치형 데이터를 범주형으로 변환할지 여부는 실제 데이터와 문제의 특성, 그리고 도메인 지식을 바탕으로 결정해야 합니다. 실험을 통해 성능 변화를 관찰하고, 최적의 전처리 전략을 선택하는 것이 좋습니다.



새로운 feature를 만들 때 좋은 방법 몇가지만 알려줘

새로운 특성(feature)을 생성하는 과정을 '특성 공학(feature engineering)'이라고 합니다. 특성 공학은 모델의 성능을 크게 향상시킬 수 있는 중요한 단계 중 하나입니다. 다음은 새로운 특성을 생성하는 몇 가지 일반적인 방법입니다:

1. 도메인 지식 활용: 문제의 도메인 지식을 활용하여 새로운 특성을 생성합니다. 예를 들어, 소매업에서는 "평일/주말"이나 "휴일/비휴일"과 같은 정보가 중요한 특성으로 활용될 수 있습니다.
2. 상호작용 특성(Interaction features): 두 가지 이상의 특성을 결합하여 새로운 특성을 만듭니다. 예를 들어, 길이와 너비가 있으면 이 두 값을 곱하여 "면적"이라는 새로운 특성을 만들 수 있습니다.
3. 다항 특성(Polynomial features): 특성의 제곱, 세제곱 등을 새로운 특성으로 추가합니다. 이는 비선형 관계를 캡처하는 데 도움이 될 수 있습니다.
4. 통계적 집계(Aggregation): 여러 데이터 포인트를 집계하여 새로운 특성을 생성합니다. 예를 들어, 사용자별 구매 이력에서 월별 구매 횟수, 평균 구매 금액 등을 계산하여 새로운 특성으로 사용할 수 있습니다.
5. 시간 기반 특성: 날짜 및 시간 관련 특성에서 "요일", "시간대", "계절" 등의 정보를 추출하여 사용합니다.
6. 원-핫 인코딩(One-hot encoding): 범주형 변수를 이진 특성으로 변환합니다.
7. 범주형 변수의 통계치 활용: 범주형 변수를 기반으로 그룹별 평균, 중앙값, 표준편차 등의 통계치를 계산하여 새로운 특성으로 활용합니다.
8. 임베딩(Embeddings): 텍스트나 범주형 데이터를 저차원의 밀집 벡터로 변환하는 방법으로, 주로 딥러닝 모델에서 사용됩니다.
9. 이동 평균 및 시간 창(Windowed features): 시계열 데이터에서 이전 N개의 포인트에 대한 평균, 표준편차 등을 계산합니다.
10. 비율 및 차이: 기존의 두 특성 간의 비율이나 차이를 계산하여 새로운 특성으로 사용합니다.

특성을 추가할 때는 항상 모델의 성능을 검증하는 과정이 필요하며, 모든 특성이 반드시 성능 향상에 기여하는 것은 아닙니다. 따라서 새로운 특성을 추가한 후에는 항상 교차 검증(cross-validation) 등의 방법으로 성능을 평가하고 필요하다면 특성 선택(feature selection)을 통해 중요한 특성만을 유지하는 것이 좋습니다.

(권도혁) Data 분석 정리

새로운 Feature 제작 후 기존 Feature를 삭제하는 것이 좋은 영향을 주는가?

경험적으로 그렇다는 생각이 들었다. 본 대회에서 초기 접근 방법으로는 새로운 Feature를 만들면서 기존의 Feature가 더이상 의미 없다고 판단하여 삭제하였는데, 이를 이후에 삭제하지 않았을 때보다 더 낮은 점수가 나왔다.



submission.csv

Complete · Dohyeok Kwon · 2d ago

0.79705



submission.csv

Complete · Dohyeok Kwon · 2d ago

0.79611



submission.csv

Complete · Dohyeok Kwon · 2d ago

0.79003



submission.csv

Complete · Dohyeok Kwon · 2d ago

0.75496

그러나 새로운 Feature를 만들 때 이용하였던 기존 feature를 살려서 다시 학습한 결과 점수가 점점 오르는 것을 볼 수 있었다.

새로운 Feature 제작 후 기존 Feature를 삭제하는 것이 좋은 영향을 주는가?

하지만 일반적으로 Machine Learning Model을 학습시킬 때에는 이것이 올바른 선택이라고 볼 수는 없다. 특히 선형모델의 경우 각 Feature들의 값을 직접적으로 계산에 이용해야하고, Feature가 많을수록 더욱 복잡한 계산 결과를 거쳐서 옳은 Output을 내야하기 때문에, 더욱 정교한 학습 모델을 생성할 수는 있지만, 정확한 함수가 되도록 학습하는 것이 그만큼 어려워진다고 볼 수 있다.

Feature Selection (특징 선택)

From many features to a few that are useful

특징 랭킹(feature ranking) 또는 특징 중요도(feature importance)라고도 함

특징 선택의 목적은 모든 특징의 부분 집합을 선택하거나, 불필요한 특징을 제거하여 간결한 특징 집합을 만드는 것

- 분석자가 사전의 배경 지식을 이용해서 진행
- 자동 특징 선택(automatic feature selection methods)을 사용

Decision Tree의 경우는 트리의 상단에 있을수록 중요도가 높음

회귀모델의 경우 forward selection과 backward elimination 같은 알고리즘을 통해 특징을 선택

Feature Selection



출처: <https://velog.io/@guide333/%EC%95%84%EC%9D%B4%ED%9A%A8-Feature-Engineering>

[Machine Learning] Feature Engineering

Feature Engineering 모델 정확도를 높이기 위해서 주어진 데이터를 예측 모델의 문제를 잘 표현할 수 있는 features로 변형시키는 과정 머신러닝 모델을 위한 데이터 데

<https://yumdata.tistory.com/385>



위 글에서 알 수 있듯 Feature를 선택하는 것은 불필요한 feature를 제거하고, 보다 필요한 정보만을 이용해서 간단하게 원하는 결과를 내는 모델을 만들 수 있는 과정이다. 다만 본 대회에 제출했던 모델들은 feature 선택이 사전 배경 지식에 의해 이뤄졌기 때문에 중요한 정보를 온전히 담아내지 못한 feature를 만들고 기존 feature를 삭제했기 때문이라고 볼 수 있다.

기존 feature의 정보를 이용할 것이라면,

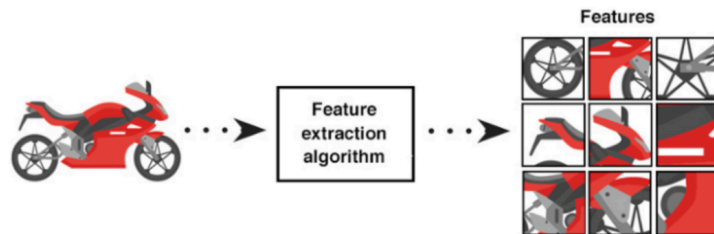
새로운 feature를 만들어내는 것이 의미가 없는 것이 아닐까?

그렇지는 않다. Random Forest를 기반으로 한 모델들의 경우 feature들 간의 다양한 조합을 통해 여러 Tree를 생성하고 이 결과들을 종합하여 판단하는 방식을 통해 output을 내기 때문에 충분한 데이터 분석과 시각화를 통해 판단한 정보로 만들어낸 feature라면 기존 feature에서 이끌어내지 못한 정보들을 보다 직접적으로 포함시켜 판단을 내릴 수 있게 되는 것이다.

PCA(Principle Component Analysis)

- 각 변수(feature)를 하나의 축으로 투영시켰을 때 분산이 가장 큰 축을 첫번째 주성분으로 선택하고 그 다음 큰 축을 두번째 주성분으로 선택하고 데이터를 선형 변환하여 다차원을 축소하는 방법

unsupervised clustering methods



출처: <https://velog.io/@guide333/%EC%95%84%EC%9D%B4%ED%9A%A8-Feature-Engineering>

특히, 각 Tree 모델들은 feature간 중요도를 판단하는 과정을 포함하는데, 한 feature안에 숨겨져있는 중요한 정보들은 feature의 우선순위 선택만으로는 쉽게 드러나지 않는다. 따라서 PCA 등의 주성분 분석 과정에서 필요한 중요한 정보만을 우선순위로 올리는 형태로 학습이 되게 하려면, 배경 지식 등을 통해 새롭게 만들어진 feature들이 필요하다고 볼 수 있다.

그렇다면 어떠한 과정으로 Feature를 만들고 선택해야 하는가?

이는 보통 'Feature Engineering Technique'에 따라 진행되는데, 그 과정은 다음과 같다.

1. Imputation (산입)

데이터 중에 missing value(결측치)를 대체값으로 채움

- Numeric Imputation: 숫자 컬럼에서는 Default 값(0이나 중앙값)으로 대체
- Categorical Imputation: 가장 빈번히 나타나는 값으로 채움, 만약 많이 발생한 값이 없다면 'Other'로 대체
- Random Imputation: 데이터셋에서 임의로 뽑은 값으로 결측치 대체
- End of Distribution Imputation: 결측치를 $\text{mean} + 3 \times \text{std}$ 로 대체

missing value(결측치)

- 머신러닝 모델 성능에 영향을 준다
- 70% 이상의 결측치가 있는 행과 열은 지워주는 것이 좋다

2. Handling Outliers (이상치 처리하기)

outliers 때문에 학습이 왜곡되는 경우가 있어 제거 필요

outliers는 시각적으로 확인 필요

표준편차를 이용한 이상치 탐지

- 표준편차에서 예외적으로 나타내는 값들을 제거
- 값과 평균까지의 거리 $> x \times \text{표준편차}$ ($x: 2 \sim 4$ 사이의 값)라면 이상치

백분위수(percentile)를 이용한 이상치 탐지

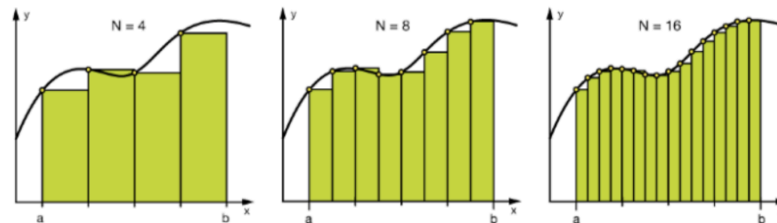
- 특정 percent외에 있는 값들을 삭제(상위 5%, 하위 5% 등)

3. Binning (Bin으로 묶기)

통계에서 Bin은 히스토그램의 하나의 막대 정도를 나타냄

근처의 값들을 하나의 범주로 묶게되면 좀 더 Robust한 모델을 얻을 수 있다.

단, 주요 정보를 잃을수도 있기 때문에 얼마만큼 하나의 Bin으로 묶을지 주의해야 한다.



Binning illustration of numerical data

출처: <https://velog.io/@guide333/%EC%95%84%EC%9D%B4%ED%9A%A8-Feature-Engineering>

categorical / numeric data에 모두 적용 가능

Bining을 하는 이유는 모델을 더 robust하게 만들고 오버피팅을 방지하기 위함

비용이 비쌈(the trade-off b/w performance and overfitting)

숫자로 이루어진 열의 경우에는 별 필요가 없지만, 범주형 열에는 필요할 수 있다.

- 예, 100,000개의 열로 구성된 데이터셋의 경우, 100개 이하로 구성된 카테고리들은 하나로 합치는 것이 나올 수 있음

출처 : <https://yumdata.tistory.com/385>

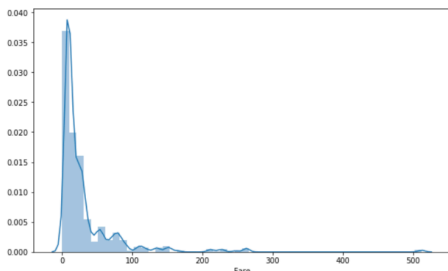
그렇다면 어떠한 과정으로 Feature를 만들고 선택해야 하는가?

4. Log Transform (로그 변형)

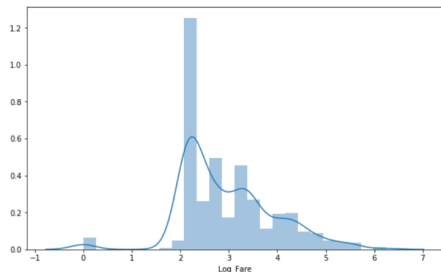
한 쪽으로 편향된(skewed) 데이터를 분포가 정규 분포에 가깝게 만들어 주는 것
이상치의 영향을 줄여주고 모델이 보다 robust하게 만들어 준다.

주의: 데이터가 양수로 이루어져야 한다. 때로 1을 더해줄 수도 있다 ($\log(x+1)$)

- 예, 나이와 재산 관계에서, 나이는 1~100까지의 범주를 갖지만, 재산은 0원에서 수조원까지 아주 넓게 분포할 수 있다.
- 특정 영역(이를 테면 1억원)에 치우쳐 있다면, 그 외의 값들은 왜곡되어 해석될 가능성이 높다.



Skewness: 0.395
Kurtosis: 0.976



출처 : <https://yumdata.tistory.com/385>

5. One Hot Encoding

텍스트로 된 범주형 데이터를 학습에 이용하기 위해서는 숫자 타입으로 변형 필요

단순 숫자 부여할 경우 순서를 넘어서 크기로 인식할 수 있기 때문에,

각각의 데이터를 하나의 컬럼으로 만들고, 해당값이 있는지를 0과 1로 표현하는 것

A B		C D E F G H I					
1 Original data:		2 One-hot encoding format:					
2 id	Color	id	White	Red	Black	Purple	Gold
3 1	White	1	1	0	0	0	0
4 2	Red	2	0	1	0	0	0
5 3	Black	3	0	0	1	0	0
6 4	Purple	4	0	0	0	1	0
7 5	Gold	5	0	0	0	0	1
8							

6. Group Operation (그룹 연산)

Tidy

- 데이터셋의 변수가 열, 관측값이 행으로 이루어진 데이터
- 조작, 모델링, 시각화가 쉬움

현실에서는 Tidy 데이터가 아닌 경우가 많다.

- 여행사의 DB의 경우 한 사용자가 여러번의 transaction을 일으킬 수 있음
- 공자의 센서데이터는 의미없는 유사한 데이터를 지속적으로 발생시킬 수 있음

모든 행들 각각이 하나의 사건 혹은 의미를 담는다면 분석이 용이해진다.

일반적으로 데이터는 변수가 행으로 이루어져 있으므로 변수에 의해 데이터를 그룹화하여 한 행으로 변수가 표현되게 만든다.

그룹화할 때 features의 병합 함수를 사용해야 한다.

- 예, 숫자로 이루어진 features는 평균과 합으로 그룹화

범주형 데이터의 경우는 pivot 테이블을 이용하거나 one-hot encoding을 한다.

User	City	Visit Days
1	Roma	1
2	Madrid	2
1	Madrid	1
3	Istanbul	1
2	Istanbul	4
1	Istanbul	3
1	Roma	3



User	Istanbul	Madrid	Roma
1		3	1
2		4	2
3	1	0	0

그렇다면 어떠한 과정으로 Feature를 만들고 선택해야 하는가?

7. Feature Split

유용한 방법. 여러 가지 방법이 존재

- 예, 이름과 성을 따로 분리

8. Scaling

대부분의 데이터가 range가 다른기 때문에 scaling을 해주어야 한다.

Normalization (min-max normalization)

- 데이터를 0과 1사이의 값으로 변환
- 데이터 분포도 변하지 않고 표준 편차가 줄어들어 outlier의 영향이 커질 수 있으므로 이를 미리 제거해야 한다.

Standardization (z-score normalization)

- 데이터마다 range가 다름
- outlier를 효과적으로 제거해 주지만, 정확히 동일한 척도로 데이터를 만들어 주지는 않는다.

9. Extracting Data

날짜를 다른 행으로 분리: 년도, 월, 요일 등

특정 날짜에 대한 feature를 뽑아냄: 주간의 이름(1주, 2주 등) 주말인지 아닌지, 공휴일인지 아닌지

이 중 일부는 Tree Model에는 의미가 없는 feature engineering 방법도 있으나, 일반적인 Machine Learning Model 학습에 있어 도움이 되는 방법들을 모아둔 것이라, 상황에 맞게 사용하면 될 것이다.