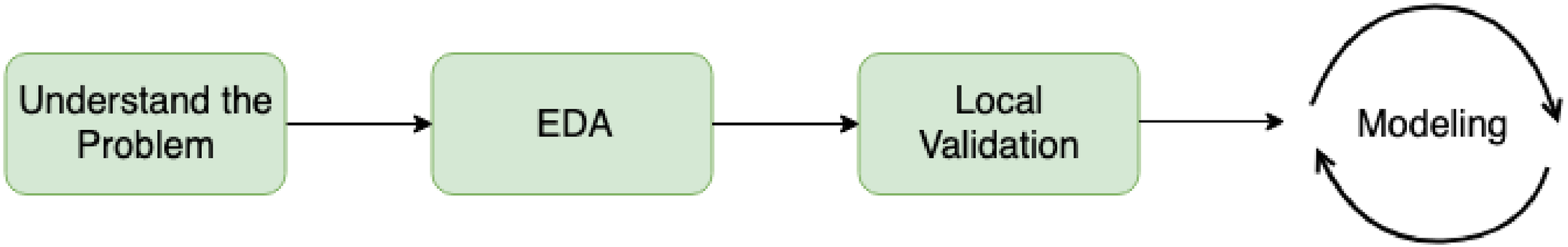
Feature engineering

WINNING A KAGGLE COMPETITION IN PYTHON

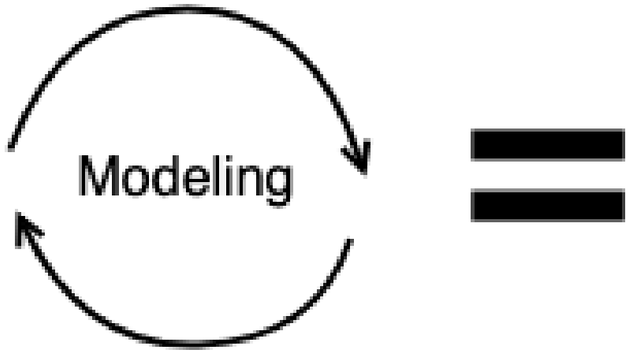
Yauhen Babakhin
Kaggle Grandmaster

Solution workflow

앞에 했던 솔루션 플로우고

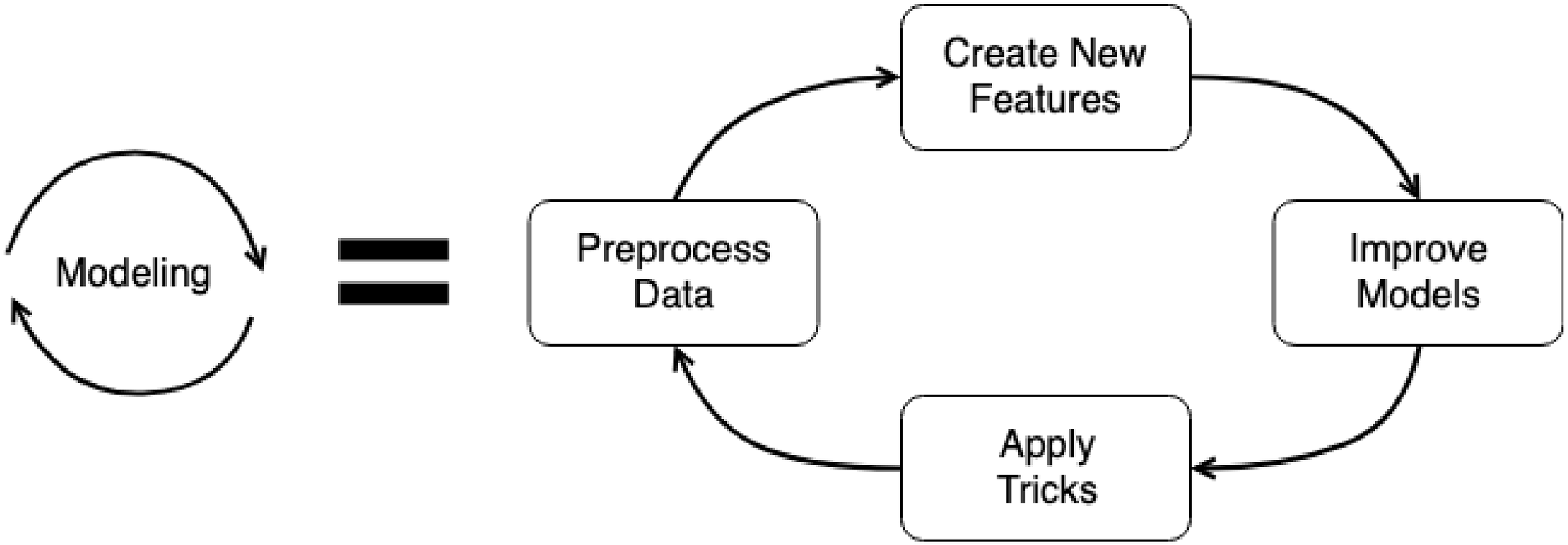


Modeling stage



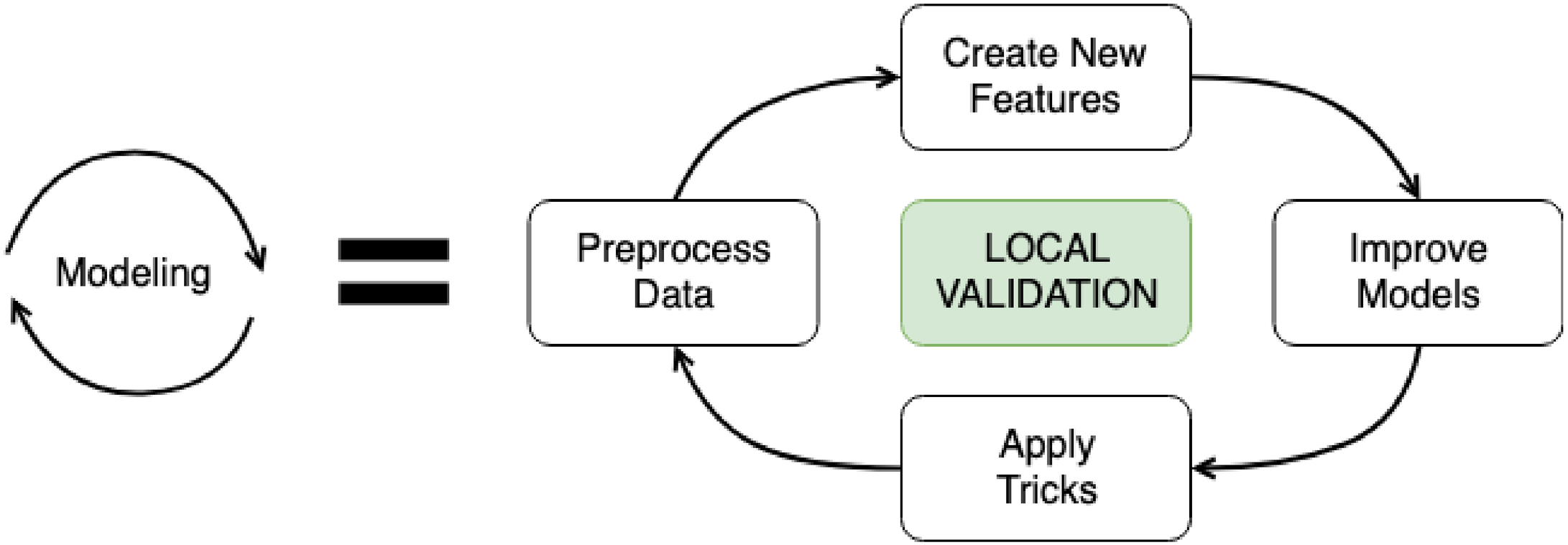
Modeling stage

모델링 루프동안 데이터를 처리하고, 새로운 feature을 만들고, 모델을 향상시키는 것을 반복해줌으로 공개 리더보드 점수를 향상시켜줍니다.



Modeling stage

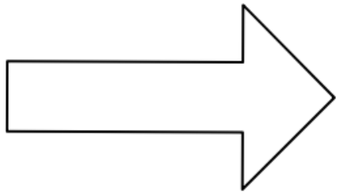
따라서 validation점수를 확인해서 개선이 되면 변경사항을 유지하고,
그렇지 않으면 버립니다. 여기서 중요한 ㄴ규칙은 한번에 하나의 것만 바꾸는 것입니다.



Feature engineering

feature engineering은 새로운 feature을 만드는 과정입니다.
모델이 추가정보를 얻고 결과적으로 목표 변수를 더 잘 예측하는데 도움을 줍니다.

Initial data

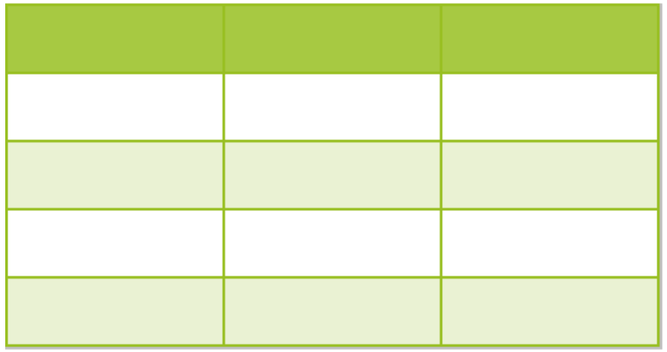


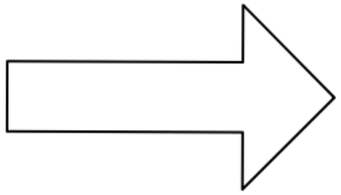
Initial data | New features

Feature engineering

새로운 feature에 대한 아이디어는 EDA나 유사한 데이터를 사용한 사전 경험에서 나올 수 있습니다.

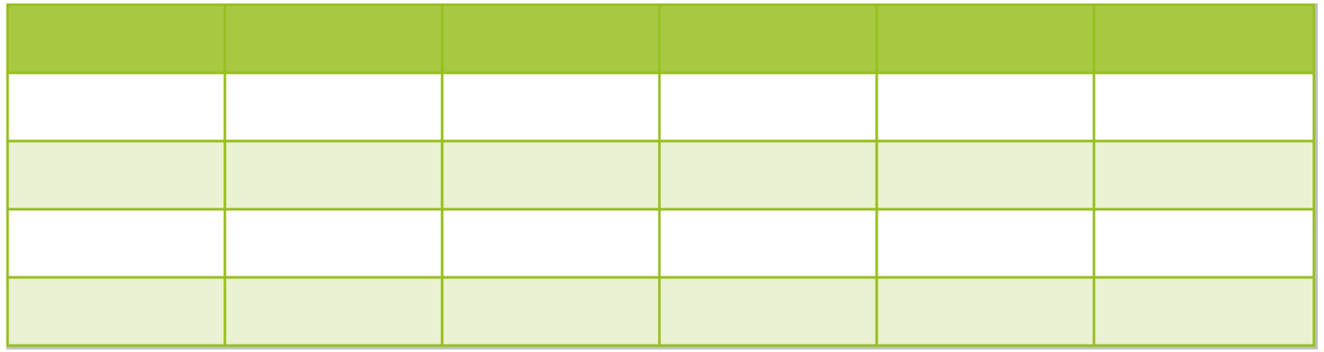
Initial data





Prior experience
EDA
Domain knowledge

Initial data | New features



Feature types

feature에는 다양한 종류가 있습니다.

- Numerical
- Categorical
- Datetime
- Coordinates
- Text
- Images

Creating features

```
# Concatenate the train and test data
data = pd.concat([train, test])
```

concat을 사용하여 train 및 test frame을 단일 데이터로 합쳐주고
그런 다음 새로운 feature을 생성해줍니다.
isin 메서드를 사용하여 각 원래 train과 테스트 ID를 찾을 수 있습니다.

```
# Create new features for the data DataFrame...
```

```
# Get the train and test back
train = data[data.id.isin(train.id)]
test = data[data.id.isin(test.id)]
```

Arithmetical features

```
# Two sigma connect competition
two_sigma.head(1)
```

단순한 공학적인 feature들은 산술적 feature들이 있는데
두 가지 수치적 특징을 취하고, 산술적 연산을 적용하여 새로운 feature을 얻습니다.

```
      id     bathrooms  bedrooms price interest_level
0     10     1.5        3        3000  medium
```

```
# Arithmetical features
two_sigma['price_per_bedroom'] = two_sigma.price / two_sigma.bedrooms
two_sigma['rooms_number'] = two_sigma.bedrooms + two_sigma.bathrooms
```

Datetime features

```
# Demand forecasting challenge
dem.head(1)
```

```
          id            date store  item  sales
0     100000     2017-12-01     1     1     19
```

```
# Convert date to the datetime object
dem['date'] = pd.to_datetime(dem['date'])
```

날짜에서 feature을 생성하려면 먼저 판다의 to_datetime() 방법을 사용하여 날짜 열을 datetime 객체로 변환한다.

Datetime features

.dt 속성을 사용하여 원하는 날짜 feature을 얻을 수 있습니다.

```
# Year features
dem['year'] = dem['date'].dt.year
# Month features
dem['month'] = dem['date'].dt.month
# Week features
dem['week'] = dem['date'].dt.weekofyear
```

date	year	month	week
2017-12-01	2017	12	48
2017-12-02	2017	12	48
2017-12-03	2017	12	48
2017-12-04	2017	12	49

```
# Day features
dem['dayofyear'] = dem['date'].dt.dayofyear
dem['dayofmonth'] = dem['date'].dt.day
dem['dayofweek'] = dem['date'].dt.dayofweek
```

date	dayofyear	dayofmonth	dayofweek
2017-12-01	335	1	4
2017-12-02	336	2	5
2017-12-03	337	3	6
2017-12-04	338	4	0

Let's practice!

WINNING A KAGGLE COMPETITION IN PYTHON

Categorical features

WINNING A KAGGLE COMPETITION IN PYTHON

Yauhen Babakhin
Kaggle Grandmaster

Label encoding

ID	Categorical feature
1	A
2	B
3	C
4	A
5	D
6	A

ID	Label-encoded
1	0
2	1
3	2
4	0
5	3
6	0

슬라이드의 범주형 피쳐의 예를 생각해 보면 대부분 기계학습 모델은 문자열 값과 범주형 feature을 자동으로 처리해주지 않기 때문에 숫자로 변경해줘야합니다.

Label encoding

```
# Import LabelEncoder
from sklearn.preprocessing import LabelEncoder
# Create a LabelEncoder object
le = LabelEncoder()
# Encode a categorical feature
df['cat_encoded'] = le.fit_transform(df['cat'])
```

```
   ID  cat  cat_encoded
0   1    A            0
1   2    B            1
2   3    C            2
3   4    A            0
```

One-Hot encoding

ID	Categorical feature
1	A
2	B
3	C
4	A
5	D
6	A

ID	Cat == A	Cat == B	Cat == C	Cat == D
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	1	0	0	0
5	0	0	0	1
6	1	0	0	0

encoding을 해서 해당 범주가 아닌 것들은 0으로 설정합니다.

One-Hot encoding

```
# Create One-Hot encoded features
ohe = pd.get_dummies(df['cat'], prefix='ohe_cat')
# Drop the initial feature
df.drop('cat', axis=1, inplace=True)
# Concatenate OHE features to the dataframe
df = pd.concat([df, ohe], axis=1)
```

판다스에서는 get_dummies를 이용해서 원핫 인코딩을 해줍니다.

	ID	ohe_cat_A	ohe_cat_B	ohe_cat_C	ohe_cat_D
0	1	1	0	0	0
1	2	0	1	0	0
2	3	0	0	1	0
3	4	1	0	0	0

Binary Features

```
# DataFrame with a binary feature
binary_feature
```

이진 피처인 경우 항상 레이블 인코딩을 적용하여 첫번째 범주를 0으로, 두번째 범주를 1로 대체합니다.

```
      binary_feat
0     Yes
1     No
```

```
le = LabelEncoder()
binary_feature['binary_encoded'] = le.fit_transform(binary_feature['binary_feat'])
```

```
  binary_feat binary_encoded
0     Yes     1
1     No      0
```

Other encoding approaches

다른 범주형 피쳐 인코더의 목록이 있습니다.

- Backward Difference Coding
- BaseN
- Binary
- CatBoost Encoder
- Hashing
- Helmert Coding
- James-Stein Encoder
- Leave One Out
- M-estimate
- One Hot
- Ordinal
- Polynomial Coding
- Sum Coding
- Target Encoder
- Weight of Evidence

Other encoding approaches

Kaggle에서 가장 널리 사용되는 것은 타겟 인코더입니다.

- Backward Difference Coding
- BaseN
- Binary
- CatBoost Encoder
- Hashing
- Helmert Coding
- James-Stein Encoder
- Leave One Out
- M-estimate
- One Hot
- Ordinal
- Polynomial Coding
- Sum Coding
- **Target Encoder**
- Weight of Evidence

Let's practice!

WINNING A KAGGLE COMPETITION IN PYTHON

Target encoding

WINNING A KAGGLE COMPETITION IN PYTHON

Yauhen Babakhin
Kaggle Grandmaster

High cardinality categorical features

높은 카디널리티는 매우 드물거나 고유한 값이 있는 열을 나타냅니다.

- Label encoder provides distinct number for each category
- One-hot encoder creates new feature for each category value
- **Target encoding to the rescue!**

높은 카디널리티의 경우, 순서가 매겨지지 않은 정수 숫자가 많은 feature을 갖게 된다는 것을 의미합니다.

Mean target encoding

Train ID	Categorical	Target
1	A	1
2	B	0
3	B	0
4	A	1
5	B	0
6	A	0
7	B	1

Test ID	Categorical	Target
10	A	?
11	A	?
12	B	?
13	A	?

인코딩에는 다양한 옵션이 있는데, 여기서는 kaggle에서 가장 자주 사용되는 평균 대상 인코딩을 볼것입니다.

Mean target encoding

1. Calculate mean on the train, apply to the test
2. Split train into K folds. Calculate mean on (K-1) folds, apply to the K-th fold
3. Add mean target encoded feature to the model

특정 feature에 평균 대상 인코딩을 적용하려면
먼저, train data의 각 범주에 대한 평균 목표값을 계산하고
그런 다음 다음 이러한 통계를 test 데이터의 해당 범주에 적용합니다.
그런 후 train data를 k개의 fold로 나누고 특정 접힌것?들을 제외한 것의 목표 평균을 계산합니다.
이렇게 하면 train set에 과적합하는 것을 막을 수 있습니다.

Calculate mean on the train

Train ID	Categorical	Target
1	A	1
2	B	0
3	B	0
4	A	1
5	B	0
6	A	0
7	B	1

test data의 범주를 인코딩하기 위해 전체 train data를 가져와서 각 범주에 대한 평균 목표 값을 계산합니다.

Calculate mean on the train

Train ID	Categorical	Target
1	**A**	**1**
2	B	0
3	B	0
4	**A**	**1**
5	B	0
6	**A**	**0**
7	B	1

범주 A의 경우 0.66(3개의 관측치 중 2개의 양수 값)과 같습니다.

Calculate mean on the train

Train ID	Categorical	Target
1	A	1
2	B	0
3	B	0
4	A	1
5	B	0
6	A	0
7	B	1

B의 경우 0.25(4개의 관측치 중 1개의 양수 값)와 같습니다.

Test encoding

Test ID	Categorical	Target	Mean encoded
10	A	?	0.66
11	A	?	0.66
12	B	?	0.25
13	A	?	0.66

이러한 통계는 테스트 데이터의 해당 범주에 적용됩니다.

Train encoding using out-of-fold

Train ID	Categorical	Target	Fold
1	A	1	1
2	B	0	1
3	B	0	1
4	A	1	1
5	B	0	2
6	A	0	2
7	B	1	2

이제 train 데이터에 대한 평균 대상 인코딩 기능을 계산해야 하는데 2개로 나누어줍니다.

Train encoding using out-of-fold

Train ID	Categorical	Target	Fold	Mean encoded
1	A	1	1	
2	B	0	1	
3	B	0	1	
4	A	1	1	
5	**B**	**0**	2	
6	**A**	**0**	2	
7	**B**	**1**	2	

Train encoding using out-of-fold

Train ID	Categorical	Target	Fold	Mean encoded
1	A	1	1	0
2	B	0	1	0.5
3	B	0	1	0.5
4	A	1	1	0
5	**B**	**0**	2	
6	**A**	**0**	2	
7	**B**	**1**	2	

범주 A가 0을 얻고 범주 B가 0.5를 얻는데
그 이유는 목표 평균을 계산하므로 2번 접힌 관측치만 사용하게 됩니다?

Train encoding using out-of-fold

Train ID	Categorical	Target	Fold	Mean encoded
1	**A**	**1**	1	0
2	**B**	**0**	1	0.5
3	**B**	**0**	1	0.5
4	**A**	**1**	1	0
5	B	0	2	
6	A	0	2	
7	B	1	2	

Train encoding using out-of-fold

Train ID	Categorical	Target	Fold	Mean encoded
1	**A**	**1**	1	0
2	**B**	**0**	1	0.5
3	**B**	**0**	1	0.5
4	**A**	**1**	1	0
5	B	0	2	0
6	A	0	2	1
7	B	1	2	0

첫 번째 접힘 관측치만 사용하여 두 번째 접힘에 대한 접힘 외 목표 평균을 계산합니다. 범주 A는 1을 얻고 범주 B는 0을 얻는다.

Practical guides

항상 평균 대상 인코딩과 함께 적용되는 몇 가지 실용적인 팁에는

Practical guides

Smoothing

$$mean\_enc_i = \frac{target\_sum_i}{n_i}$$

$$smoothed\_mean\_enc_i = \frac{target\_sum_i + \alpha * global\_mean}{n_i + \alpha}$$

$$\alpha \in [5; 10]$$

첫번째 스무딩이 있습니다.
특정 범주에 대해 평균을 취해주게되면 값이 하나 또는 두개뿐인 범주는 과적합으로 이어질 수 있습니다.
그러므로 먼저 global mean(전체 train data에 대한 목표 평균 값)을 계산해준 후 전역 평균을 사용하여 알파를 각 범주에 추가해줍니다.
이제 범주가 크면 평균 인코딩을 신뢰하고 그렇지 않으면 글로벌 평균을 고수합니다.

Practical guides

Smoothing

$$mean\_enc_i = \frac{target\_sum_i}{n_i}$$

$$smoothed\_mean\_enc_i = \frac{target\_sum_i + \alpha * global\_mean}{n_i + \alpha}$$

$$\alpha \in [5; 10]$$

New categories

- Fill new categories in the test data with a *global_mean*

또 다른 방법은 시험 데이터의 새로운 범주에 관한 것인데, 이 경우 범주의 목표 평균값이 무엇인지 알수 없으므로
새 범주 값은 단순히 목표 전역 평균으로 채워집니다.

Practical guides

Train ID	Categorical	Target
1	A	1
2	B	0
3	B	0
4	A	0
5	B	1

Test ID	Categorical	Target	Mean encoded
10	A	?	0.43
11	B	?	0.38
12	C	?	0.40

그 예시를 보세요. 초기 설정에서 범주 A는 0.5, 범주 B는 1/3을 얻을 것이다. 그러나 알파가 5인 경우 범주 A는 약 0.43이고 범주 B는 약 0.38입니다. 검정 데이터의 새 범주 C가 전역 평균을 얻는 반면, 이는 0.4와 같습니다.

Let's practice!

WINNING A KAGGLE COMPETITION IN PYTHON

Missing data

WINNING A KAGGLE COMPETITION IN PYTHON

Yauhen Babakhin
Kaggle Grandmaster

Missing data

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	NaN	1
5	NaN	2.6	0
6	A	5.3	0

Impute missing data

Numerical data

- Mean/median imputation

결측값을 채우는 가장 간단한 방법은
평균 또는 중위수로 채워줍니다.

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	NaN	1
5	NaN	2.6	0
6	A	5.3	0

Impute missing data

Numerical data

- Mean/median imputation
- Constant value imputation

이 예제에서는 결측값을 4.72로 변경합니다.
하지만, 이렇게 채우면 실제로 누락된 값이라는 정보를 잃기 때문에
데이터가 결측값임을 강조하기 위해 특수 상수값을 사용하기도 합니다.

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	**4.72**	1
5	NaN	2.6	0
6	A	5.3	0

Impute missing data

Numerical data

- Mean/median imputation
- Constant value imputation

예를 들어 -999입니다.

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	**-999**	1
5	NaN	2.6	0
6	A	5.3	0

Impute missing data

Numerical data

- Mean/median imputation
- Constant value imputation

Categorical data

- Most frequent category imputation

범주형 feature을 채워넣기 위해
데이터에서 가장 빈도가 높은 범주를 채우거

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	**-999**	1
5	NaN	2.6	0
6	A	5.3	0

Impute missing data

Numerical data

- Mean/median imputation
- Constant value imputation

Categorical data

- Most frequent category imputation
- New category imputation

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	**-999**	1
5	**A**	2.6	0
6	A	5.3	0

Impute missing data

Numerical data

- Mean/median imputation
- Constant value imputation

Categorical data

- Most frequent category imputation
- New category imputation

ID	Categorical feature	Numerical feature	Binary target
1	A	5.1	1
2	B	7.2	0
3	C	3.4	0
4	A	**-999**	1
5	**MISS**	2.6	0
6	A	5.3	0

결측값에 대한 새 범주를 만듭니다.

Find missing data

```
df.isnull().head(1)
```

```
      ID    cat    num  target
0  False  False  False   False
```

```
df.isnull().sum()
```

isnull메서드를 가지고 결측값의 정보를 얻습니다.

```
ID        0
cat       1
num       1
target    0
```

Numerical missing data

strategy : 채울 값에 대한 함수로 디폴트 파라미터 값은 mean 즉, 산술평균값이 됩니다. 필요시 "median"(중앙값), "most_frequent"(최빈값 : 가장 많이 나온 값), "constant"(따로 지정한 값) 으로 사용할 수 있습니다.

```
# Import SimpleImputer
from sklearn.impute import SimpleImputer
# Different types of imputers
mean_imputer = SimpleImputer(strategy='mean')
constant_imputer = SimpleImputer(strategy='constant', fill_value=-999)
# Imputation
df[['num']] = mean_imputer.fit_transform(df[['num']])
```

sklearn에서 SimpleImputer module을 사용하여 결측값을 채워넣을 수 있습니다.
여기서는 -999를 채우기 값으로 정합니다.

Categorical missing data

```
# Import SimpleImputer
from sklearn.impute import SimpleImputer


# Different types of imputers
frequent_imputer = SimpleImputer(strategy='most_frequent')
constant_imputer = SimpleImputer(strategy='constant', fill_value='MISS')
# Imputation
df[['cat']] = constant_imputer.fit_transform(df[['cat']])
```

앞에서와 비슷하게 결측값은 Miss로 표시합니다.

Let's practice!

WINNING A KAGGLE COMPETITION IN PYTHON