1. regression metrics

MSE

- Mean square error
  - optimal constant : mean

$$\frac{1}{N}\sum(y_i - \hat{y}_i)^2$$

- RMSE
  - $\sqrt{MSE}$
  - MSE는 제곱이 있으니까 루트로 원래 target과 scale을 맞춘다.
- 그렇다면 MSE, RMSE가 어떤 값을 가질 때, model이 좋다고 판단할 수 있을까?
  - baseline과 비교한다 : R-squred
- R-squared

$$1 - \frac{MSE}{\frac{1}{N}\sum(y_i - \bar{y})^2}$$

MAE

- Mean absolute error
  - MSE보다는 outlier에 덜 민감하다.
  - optimal constant : median

$$\frac{1}{N}\sum|y_i - \hat{y}_i|$$

(R)MSPE

- target이 커질수록 error의 비중을 줄일다. 예를 들어,
  - true target이 10일 때, prediction이 9
  - true target이 100일 때, prediction이 90
  - 인 경우, MSE와 MAP는 후자에 더 큰 영향을 받게 된다.
  - 이 때, 후자의 영향을 줄이고 싶으면 MSPE, MAPE를 사용하는 것이다.
- weighted MSE인 것이다.
- optimal constant : weighted target mean

$$\frac{C}{N}\sum(\frac{y_i - \hat{y}_i}{y_i})^2$$

(R)MAPE

- optimal constant : weighted target median

$$\frac{C}{N}\sum|\frac{y_i - \hat{y}_i}{y_i}|$$

(R)MSLE

- mean square logarithmic error
- root 버전이 더 많이 쓰인다.
- MSPE, MAPE와 비슷하다.
  - 하지만 log함수의 모양을 생각해보면 true target보다 크게 예측하는지 작게 예측하는지에 따라 차이가 있다.

    - 위에서 배웠던 다른 error function들은 대칭!
- target값에 log를 취한것이다.
    - +1은 log값이 0이 되지 않도록 하기 위해

$$\sqrt{\frac{1}{N}\sum(\log(y_i+1)-\log(\hat{y}_i+1))^2}$$
$$=\sqrt{MSE(\log(y_i+1),log(\hat{y}_i+1))}$$

bias 비교

- MSE는 차이가 큰 값에 biased
- MAE는 MSE보다 덜 biased
- MSPE와 MAPE는 작은 값에 biased
- MSPE, MAPE보다 덜 biased
- 상황마다 필요한 function은 다르다. 적절하게 선택하라.

2. classification metrics

Accuracy

- optimal constant : 가장 많은 class로 분류
- hard prediction

$$\frac{1}{N}\sum[y_i=\hat{y}_i]$$

logloss

- soft prediction
- optimal constant : 각 class의 비율을 사용
- $\hat{y}$은 확률
- binary

$$-\frac{1}{N}\sum y_i\log(\hat{y}_i)+(1-y_i)\log(1-\hat{y}_i)$$

- multiclass (L class)

$$-\frac{1}{N}\sum^{N}\sum^{L}y_{il}\log(\hat{y}_{il})$$

AUC ROC

- only for binary task
- ROC curve의 아래 면적
- 각 obs마다 확률을 구한 뒤에 threshold를 모두 고려한다.
- random prediction의 경우, 0.5의 값을 가진다.
- true와 pred의 순서(order)가 잘 맞아떨어지는 것을 목표로 한다.

Cohen's kappa

- binary의 경우
    - p_e : what accuracy would be on average, if we randomly permute our predictions

$$p_e=\frac{1}{N^2}\sum n_{k1}n_{k2}$$

    - 예를 들어
        - true : 고양이 10, 개 90
        - pred : 고양이 20, 개 80
        - 0.2 * 0.1 + 0.8 * 0.9 = 0.74

$$Cohen's\ Kappa = 1 - \frac{1 - accuracy}{1 - p_e} = 1 - \frac{error}{baseline\ error}$$

weight

- 각 label마다 weight를 주는 방법도 있다.

3. Loss and metric

- (target) metric : what we want to optimize
- (optimization) loss : what model optimize

4. target metric optimization

- model에서 바로 optimization 가능
    - metric = loss 의 의미
    - MSE, Logloss
- preprocess train and optimize another metric
    - 바로 optimize는 못하고 model에 맞춰서 뭔가 처리해줘야 한다.
    - MSPE, MAPE, RMSLE
- optimize another metric, postprocess prediction
    - Accuracy, Kappa
- write custom loss function
    - Any, if you can
- optimize another metric, use early stopping
    - Any

early stopping

- metric이 가장 best일 때 멈춘다.
- 기준을 optimize loss로 하는 것이 아니라!

regression metrics optimization

- MSE 일반적인 model에 다 사용
    - L2loss
- MAE
    - XGBoost, sklearn.regression은 안된다고 한다.
    - L1loss, median quantile loss
    - 0에서 미분불가니까 Huber loss를 대신 사용하기도 한다.
    - MSPE, MAPE
        - 위의 loss와는 다르게 directly optimization은 어렵다.
        - 그래서 sample_weights를 사용하면 된다.
            - 주로 model에 구현되어 있다.
            - model에 구현이 되어 있지 않으면
                - train set을 resample하면 된다.
                - `df.sample(weights=sample_weights)`

        - 그리고 MSE(MAE)를 optimize하면 된다.
    - RMSLE
        - train
            - 1. transform target : $z_i = \log(y_i + 1)$
            - 2. Fit a model with MSE loss
        - test
            - transform predictions back : $\hat{y}_i = exp(\hat{z}_i) - 1$

classification metrics optimization

- logloss
    - 쓰면 된다. 가장 많이 쓰인다.
    - probability calibration
        - platt scaling : prediction에 대해 logistic regression
        - Isotonic regression : prediction에 대해 Isotonic regression
        - Stacking : prediction에 대해 XGBoost 또는 neural net
- accuracy
    - 직접 optimize할수는 없다.
    - 다른 loss사용하고 threshold를 이용하여 accuracy를 조절한다.
- AUC
    - 수학적으로는 AUC를 바로 계산하기는 어렵지만 대부분의 model에는 다른 방법을 이용하여 구현이 되어 있다.

5. Mean encodings

- target을 이용하여 feature를 만든다.
- val, test set을 제외하고 train set으로만 이용해야 한다.
- encoding이 target과 상관관계가 있게하니까 더 좋은 성능을 보인다.
- 방법들 (binary)
    - $likelihood = \frac{C_1}{C_1+C_2}$
    - $Weight\ of\ Evidence = \ln(\frac{C_1}{C_2}) * 100$
    - $Count = C_1 = sum(target)$
    - $Diff = C_1 - C_2$
- overfitting의 위험이 있겠구나!

mean encoding regularization

- CV loop inside train set
    - robust and intuitive
    - train, val set으로 나누고 train의 target mean encoding으로 val을 채운다.
    - 그래도 여전히 biased되어 있고 leakage는 있다.
- Smoothing
    - $\frac{mean(target) * N + globalmean * \alpha}{N + \alpha}$
    - alpha로 regularization 정도를 정한다.
- Adding random noise
    - noise정도는? 어렵다. 잘찾자!
- Sorting and calculating expanding mean
    - least amount of leakage
    - no hyperparameter
    - irregular encoding quality
    - built in CatBoost

```
cumsum = df_tr.groupby(col)['target'].cumsum() - df_tr['target']
cumcnt = df_tr.groupby(col).cumcount()
train_new[col+"_mean_target"] = cumsum / cumcnt
```

Gerneralization and extension

regression and multiclass

- regression에서는 다양한 방법을 사용할 수 있다.
  - percentile, std 등등
- one vs all 이용
- time series
  - rolling statistics of target variable 등 다양한 방법 존재하겠구나.
- interaction and numerical features
  - model을 분석(DT, linear model 등등)
  - numeric을 binning하고 interactioin을 선택한다. (categorical 끼리의 interaction을 말한다.)
  - catboost에 내장되어 있는 기능 중 하나