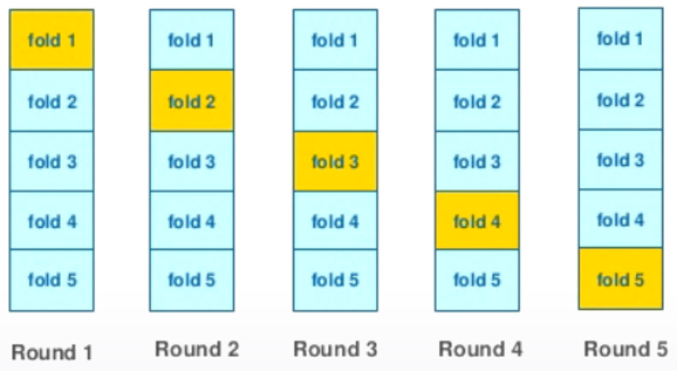
Regularization

doing a cross-validation loop to construct mean encodings

1. CV loop inside training data;
2. Smoothing;
3. Adding random noise;
4. Sorting and calculating expanding mean.

1\.

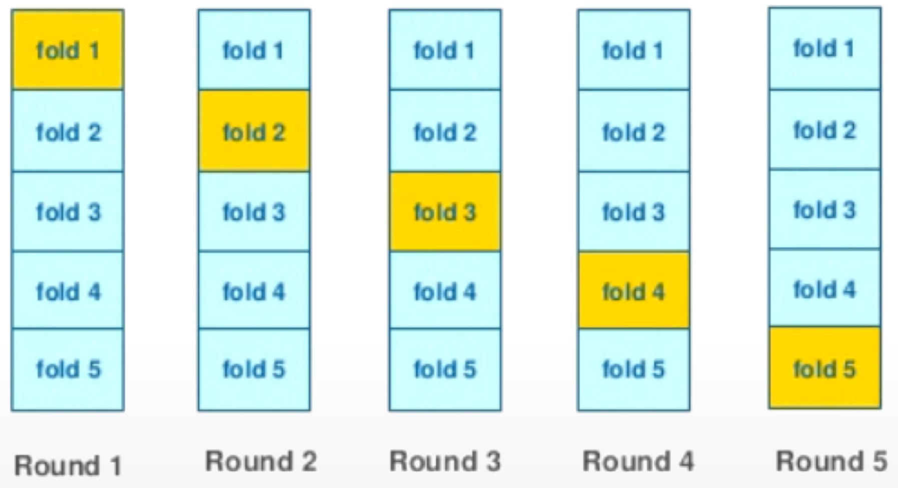


2. $$\frac{mean(target) * nrows + globalmean * alpha}{nrows + alpha}$$

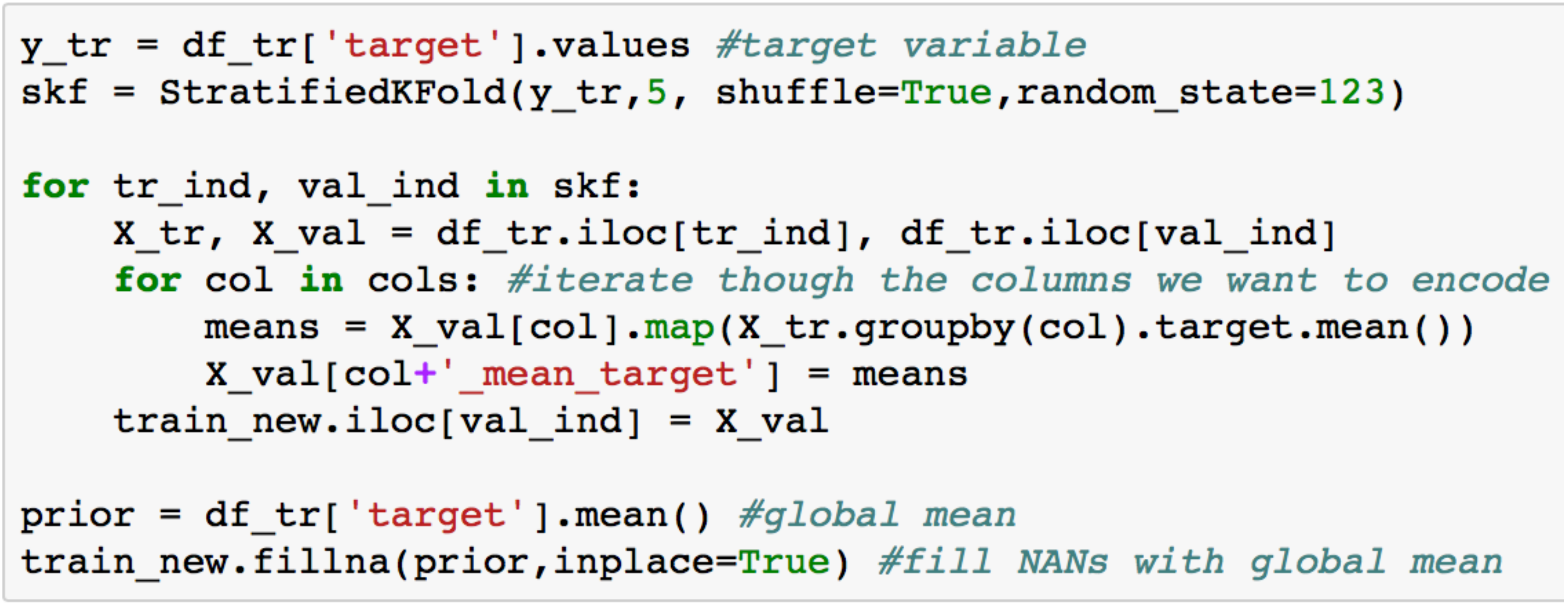
Regularization. CV loop

- Robust and intuitive
- Usually decent results with 4-5 folds across different datasets
- Need to be careful with extreme situations like LOO

KFold scheme



Regularization. CV loop



```
y_tr = df_tr['target'].values #target variable
skf = StratifiedKFold(y_tr,5, shuffle=True,random_state=123)

for tr_ind, val_ind in skf:
    X_tr, X_val = df_tr.iloc[tr_ind], df_tr.iloc[val_ind]
    for col in cols: #iterate though the columns we want to encode
        means = X_val[col].map(X_tr.groupby(col).target.mean())
        X_val[col+'_mean_target'] = means
    train_new.iloc[val_ind] = X_val

prior = df_tr['target'].mean() #global mean
train_new.fillna(prior,inplace=True) #fill NANs with global mean
```

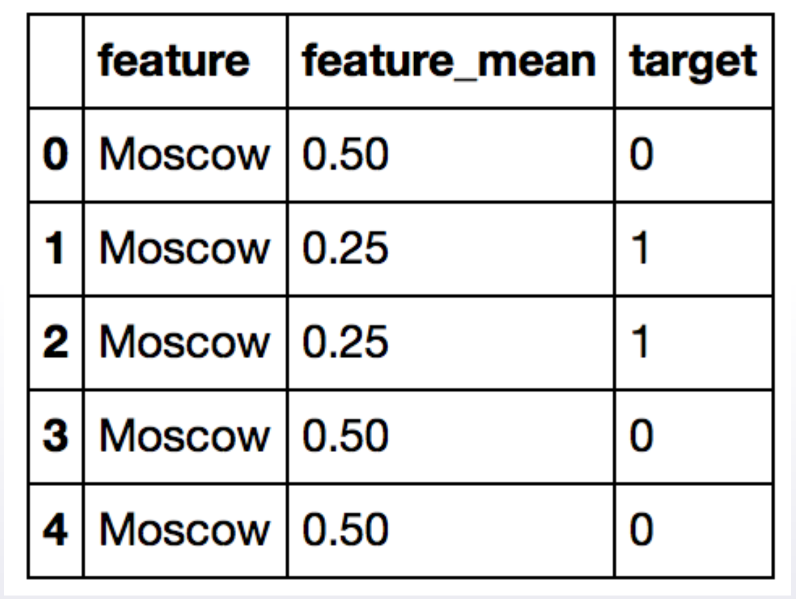
Regularization. CV loop

leave one out

- Perfect feature for LOO scheme
- Target variable leakage is still present even for KFold scheme

We didn't explicitly use target
variable, but our encoding is
biased. Furthermore, this effect remains valid even for the
KFold scheme.

Leave-one-out



	feature	feature_mean	target
0	Moscow	0.50	0
1	Moscow	0.25	1
2	Moscow	0.25	1
3	Moscow	0.50	0
4	Moscow	0.50	0

So is this type of regularization useless?

Definitely not. In practice, if you have enough data and use
four or five folds, the encodings will work fine with this
regularization strategy. Just be careful and use correct
validation.

Regularization.Smoothing

- Alpha controls the amount of regularization
- Only works together with some other regularization method

$$\frac{mean(target) * nrows + globalmean * alpha}{nrows + alpha}$$

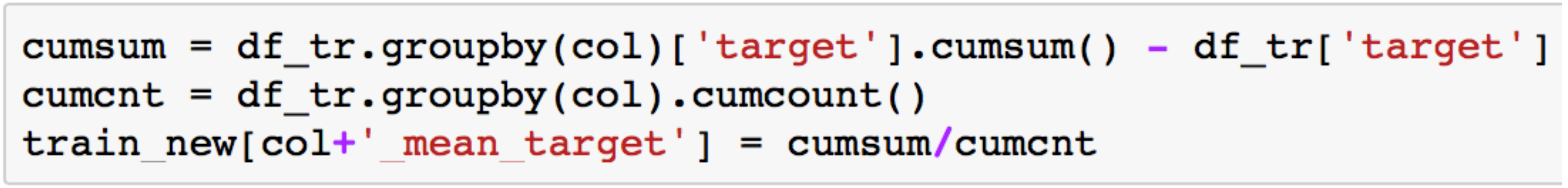
Smoothing obviously won't work on its own but we can combine it with for example, CD loop regularization.

Regularization. Noise

- Noise degrades the quality of encoding
- How much noise should we add?
- Usually used together with LOO

Regularization. Expanding mean

- Least amount of leakage
- No hyper parameters
- Irregular encoding quality
- Built - in 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
```

This method introduces the least amount of leakage from target variable and it requires no hyper parameter tuning. The only downside is that feature quality is not uniform.

Regularization. Conclusion

- There are a lot ways to regularize mean encodings
- Unending battle with target variable leakage
- CV loop or Expanding mean for practical tasks