

# **Regression metrics optimization**

# Metrics optimization: our plan

## 1) Regression

- MSE, (R)MSE, R-squared
- MAE
- (R)MSPE, MAPE
- (R)MSLE

## 2) Classification:

- Accuracy
- Logloss
- AUC
- Cohen's Kappa

# RMSE, MSE, R-squared

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

How do you optimize them?

**Just fit the right model!**

$$\text{RMSE} = \sqrt{\text{MSE}} \qquad R^2 = 1 - \frac{\text{MSE}}{\frac{1}{N} \sum_{i=1}^N (y_i - \bar{y})^2}$$

# RMSE, MSE, R-squared

- **Tree-based**

- XGBoost, LightGBM

- `sklearn.RandomForestRegressor`

- **Linear models**

- `sklearn.<>Regression`

- `sklearn.SGDRegressor`

- Vowpal Wabbit (*quantile loss*)

- **Neural nets**

- PyTorch, Keras, TF, etc.

*Synonyms: L2 loss*

Read the docs!

# MAE

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$

How do you optimize it?

**Again, just run the right model!**

# MAE

- **Tree-based**

- ~~XGBoost~~, LightGBM

- sklearn.RandomForestRegressor

- **Linear models**

- ~~sklearn.<>Regression~~

- ~~sklearn.SGDRegressor~~

- Vowpal Wabbit (*quantile loss*)

- **Neural nets**

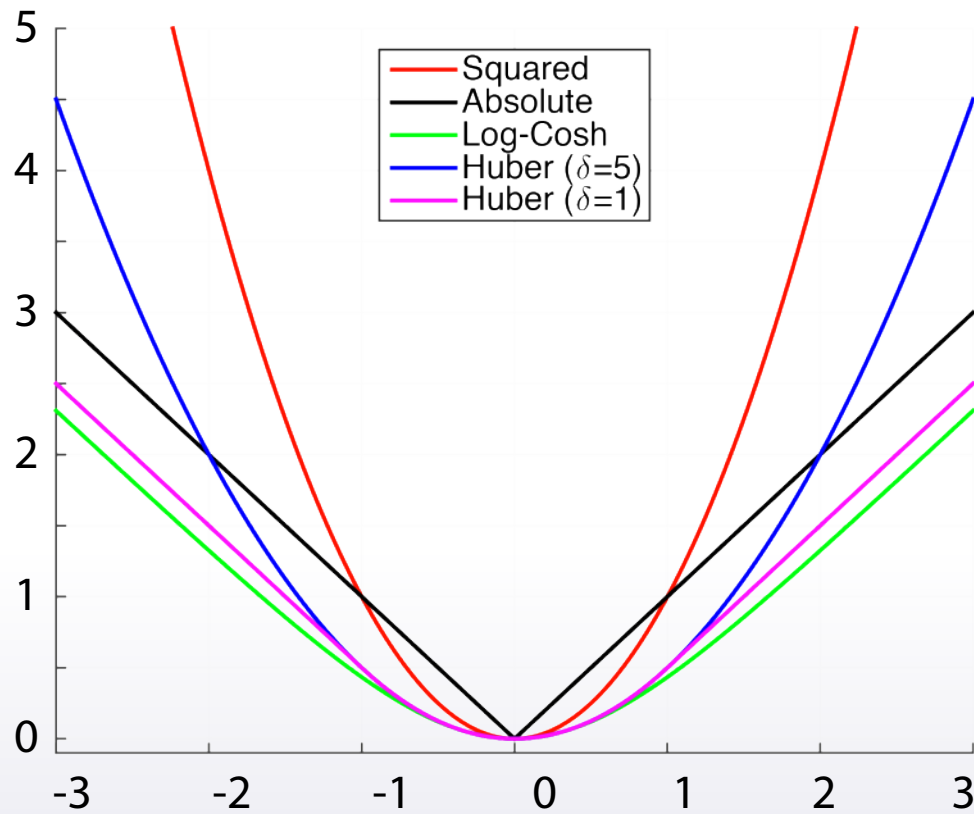
- PyTorch, Keras, TF, etc.

*Synonyms: L1, Median regression*

Read the docs!

# MAE: optimal constant

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N |y_i - \hat{y}_i|$$



# MSPE and MAPE

$$\text{MSPE} = \frac{100\%}{N} \sum_{i=1}^N \left( \frac{y_i - \hat{y}_i}{y_i} \right)^2 \quad \text{MAPE} = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

How do you optimize them?

~~Just run the right model!~~



# MSPE (MAPE) as weighted MSE (MAE)

Sample weights

$$\text{MSPE} = \frac{100\%}{N} \sum_{i=1}^N \left( \frac{y_i - \hat{y}_i}{y_i} \right)^2 \quad w_i = \frac{1/y_i^2}{\sum_{i=1}^N 1/y_i^2}$$

$$\text{MAPE} = \frac{100\%}{N} \sum_{i=1}^N \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad w_i = \frac{1/y_i}{\sum_{i=1}^N 1/y_i}$$

# MSPE (MAPE)

- **Use weights for samples (`sample\_weights`)**
  - And use MSE (MAE)

# MSPE (MAPE)

- **Use weights for samples (`sample\_weights`)**
  - And use MSE (MAE)
  - *Not every library accepts sample weights*
    - XGBoost, LightGBM accept
    - Neural nets
      - Easy to implement if not supported

# MSPE (MAPE)

- **Use weights for samples (`sample\_weights`)**
  - And use MSE (MAE)
  - *Not every library accepts sample weights*
    - XGBoost, LightGBM accept
    - Neural nets
      - Easy to implement if not supported
- **Resample the train set**
  - `df.sample(weights=sample_weights)`
  - And use *any* model that optimizes MSE (MAE)

# MSPE (MAPE)

- **Use weights for samples (`sample\_weights`)**
  - And use MSE (MAE)
  - *Not every library accepts sample weights*
    - XGBoost, LightGBM accept
    - Neural nets
      - Easy to implement if not supported
- **Resample the train set**
  - `df.sample(weights=sample_weights)`
  - And use *any* model that optimizes MSE (MAE)
  - Usually need to resample many times and average

# RMSLE

$$\begin{aligned}\text{RMSLE} &= \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\hat{y}_i + 1))^2} = \\ &= \sqrt{MSE(\log(y_i + 1), \log(\hat{y}_i + 1))}\end{aligned}$$

**Train:**

1. Transform target:

$$z_i = \log(y_i + 1)$$

2. Fit a model with MSE loss

**Test:**

- Transform predictions back:

$$\hat{y}_i = \exp(\hat{z}_i) - 1$$

# Conclusion

## 1) Regression

- MSE, (R)MSE, R-squared
- MAE
- (R)MSPE, MAPE
- (R)MSLE

## 2) Classification:

- Accuracy
- Logloss
- AUC
- Cohen's Kappa