Experiment Log

1. Encoding

a. Target Encoding sebelum split

```
...
```

	device_brand	scr
0	4.665257	
1	4.665257	
2	4.665257	
3	4.665257	
4	4.665257	

```python
brand_mean_price = df.groupby('device_brand')['normalized_used_price'].mean()
df['device_brand'] = df['device_brand'].map(brand_mean_price)
```

```python
evaluate_model(y_test, y_pred_baseline_LR, "Baseline Linear Regression")
evaluate_model(y_test, y_pred_baseline_DT, "Baseline Decision Tree")
```

```
===== Baseline Linear Regression =====
R2   : 0.836476
MAE  : 0.184911
RMSE : 0.230362
MAPE : 0.043576

===== Baseline Decision Tree =====
R2   : 0.703730
MAE  : 0.238735
RMSE : 0.310073
MAPE : 0.055480
```

```python
evaluate_model(y_test, ridge_pred, "Ridge Regression (Improved)")
evaluate_model(y_test, lasso_pred, "Lasso Regression (Improved)")
```

```
===== Ridge Regression (Improved) =====
R2   : 0.818892
MAE  : 0.192447
RMSE : 0.242431
MAPE : 0.045266

===== Lasso Regression (Improved) =====
R2   : 0.818821
MAE  : 0.193580
RMSE : 0.242478
MAPE : 0.045615
```

```
evaluate_model(y_test, poly_pred, "Polynomial Linear Regression")

===== Polynomial Linear Regression =====
R2   : 0.852128
MAE  : 0.174658
RMSE : 0.219060
MAPE : 0.040829
```

b. Target Encoding Setelah Split

```
evaluate_model(y_test, y_pred_baseline_LR, "Baseline Linear Regression")
evaluate_model(y_test, y_pred_baseline_DT, "Baseline Decision Tree")

===== Baseline Linear Regression =====
R2   : 0.836402
MAE  : 0.184987
RMSE : 0.230414
MAPE : 0.043594

===== Baseline Decision Tree =====
R2   : 0.698919
MAE  : 0.240462
RMSE : 0.312580
MAPE : 0.055929
```

```
evaluate_model(y_test, ridge_pred, "Ridge Regression (Improved)")
evaluate_model(y_test, lasso_pred, "Lasso Regression (Improved)")

===== Ridge Regression (Improved) =====
R2   : 0.818836
MAE  : 0.192524
RMSE : 0.242468
MAPE : 0.045284

===== Lasso Regression (Improved) =====
R2   : 0.818778
MAE  : 0.193640
RMSE : 0.242508
MAPE : 0.045629
```

```
===== Polynomial Linear Regression =====
R2   : 0.852135
MAE  : 0.174570
RMSE : 0.219055
MAPE : 0.040810
```

```
evaluate_model(y_test, poly_pred, "Polynomial Ridge")

===== Polynomial Ridge =====
R2   : 0.846937
MAE  : 0.177034
RMSE : 0.222871
MAPE : 0.041268
```

c. One Hot Encoding

```python
brand_dummies = pd.get_dummies(df["device_brand"],
                               prefix="brand",
                               drop_first=True)

df = pd.concat([df, brand_dummies], axis=1)
df.drop(columns=["device_brand"], inplace=True)
```

brand_Others	brand_Panasonic	brand_Realme	brand_Samsung	brand_Sony	brand_Spice	brand_Vivo	brand_XOLO	brand_Xiaomi	brand_ZTE
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

```
===== Baseline Linear Regression =====
R2   : 0.835729
MAE  : 0.184380
RMSE : 0.230888
MAPE : 0.043438

===== Baseline Decision Tree =====
R2   : 0.704020
MAE  : 0.238549
RMSE : 0.309921
MAPE : 0.055732
```

```
===== Ridge Regression (Improved) =====
R2   : 0.818510
MAE  : 0.192660
RMSE : 0.242687
MAPE : 0.045323

===== Lasso Regression (Improved) =====
R2   : 0.819229
MAE  : 0.193283
RMSE : 0.242205
MAPE : 0.045529
```

```
===== Polynomial Linear Regression =====
R2   : 0.762379
MAE  : 0.202962
RMSE : 0.277691
MAPE : 0.047509
```

d. Label Encoding

```
bool_cols = df.select_dtypes(include=['bool']).columns
df[bool_cols] = df[bool_cols].astype(int)

df.head()
```

	device_brand	screen_size	rear_camera_mp	front_camera_mp	internal_memory
0	10	14.50	13.0	5.0	64.0
1	10	17.30	13.0	16.0	128.0
2	10	16.69	13.0	8.0	128.0
3	10	25.50	13.0	8.0	64.0
4	10	15.32	13.0	8.0	64.0

Next steps: Generate code with df New interactive sheet

rice	os_Others	os_Windows	os_iOS	g4_yes	g5_yes
5100	0	0	0	1	0
9018	0	0	0	1	1
4631	0	0	0	1	1
0961	0	0	0	1	1
7837	0	0	0	1	0

```
evaluate_model(y_test, y_pred_baseline_LR, "Baseline Linear Regression")
evaluate_model(y_test, y_pred_baseline_DT, "Baseline Decision Tree")
```

```
===== Baseline Linear Regression =====
R2   : 0.836522
MAE  : 0.184795
RMSE : 0.230329
MAPE : 0.043533

===== Baseline Decision Tree =====
R2   : 0.715119
MAE  : 0.232854
RMSE : 0.304054
MAPE : 0.054451
```

```
evaluate_model(y_test, ridge_pred, "Ridge Regression (Improved)")
evaluate_model(y_test, lasso_pred, "Lasso Regression (Improved)")
```

```
===== Ridge Regression (Improved) =====
R2   : 0.818502
MAE  : 0.192737
RMSE : 0.242692
MAPE : 0.045307

===== Lasso Regression (Improved) =====
R2   : 0.818532
MAE  : 0.193749
RMSE : 0.242672
MAPE : 0.045639
```

```
===== Polynomial Linear Regression =====
R2   : 0.851843
MAE  : 0.174704
RMSE : 0.219271
MAPE : 0.040861
```

```
===== Polynomial Ridge =====
R2   : 0.845955
MAE  : 0.177436
RMSE : 0.223585
MAPE : 0.041387
```

```
===== K-FOLD RESULT : Linear Regression =====
Mean R2   : 0.839825
Mean MAE  : 0.183610
Mean RMSE : 0.237087
Mean MAPE : 0.044372
```

```python
ridge_pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("model", Ridge(alpha=1.0))
])

kfold_evaluate(
    ridge_pipeline,
    X_train,
    y_train_log,
    "Ridge Regression"
)
```

```
===== K-FOLD RESULT : Ridge Regression =====
Mean R2   : 0.821924
Mean MAE  : 0.036392
Mean RMSE : 0.049374
Mean MAPE : 0.022484
```

```
===== K-FOLD RESULT : Lasso Regression =====
Mean R2   : 0.821249
Mean MAE  : 0.036546
Mean RMSE : 0.049479
Mean MAPE : 0.022615
```

```python
poly_lr_pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("poly", PolynomialFeatures(degree=2, include_bias=False)),
    ("model", LinearRegression())
])

kfold_evaluate(
    poly_lr_pipeline,
    X_train,
    y_train,
    "Polynomial Linear Regression"
)
```

```
===== K-FOLD RESULT : Polynomial Linear Regression =====
Mean R2   : 0.846154
Mean MAE  : 0.179836
Mean RMSE : 0.232155
Mean MAPE : 0.043007
```

```
===== K-FOLD RESULT : Polynomial Ridge Regression =====
Mean R2   : 0.842588
Mean MAE  : 0.034658
Mean RMSE : 0.046329
Mean MAPE : 0.021273
```

2. Kfold evaluation (Kfold=5)

```
kfold_evaluate(
    LinearRegression(),
    X_train_scaled_df,
    y_train,
    "Linear Regression"
)
```

```
===== K-FOLD RESULT : Linear Regression =====
Mean R2   : 0.840088
Mean MAE  : 0.183548
Mean RMSE : 0.236885
Mean MAPE : 0.044354
```

```
kfold_evaluate(
    Ridge(alpha=1.0),
    X_train_scaled_df,
    y_train_log,
    "Ridge Regression"
)
```

```
===== K-FOLD RESULT : Ridge Regression =====
Mean R2   : 0.822331
Mean MAE  : 0.036354
Mean RMSE : 0.049316
Mean MAPE : 0.022459
```

```
kfold_evaluate(
    Lasso(alpha=0.001, max_iter=10000),
    X_train_scaled_df,
    y_train_log,
    "Lasso Regression"
)
```

```
===== K-FOLD RESULT : Lasso Regression =====
Mean R2   : 0.821572
Mean MAE  : 0.036528
Mean RMSE : 0.049433
Mean MAPE : 0.022602
```

3. Perbandingan KNN dengan Mean

a. Dengan KNN

```
===== Baseline Linear Regression =====
R2   : 0.836751
MAE  : 0.184613
RMSE : 0.230168
MAPE : 0.043516

===== Baseline Decision Tree =====
R2   : 0.691628
MAE  : 0.239311
RMSE : 0.316342
MAPE : 0.055914
```

```
===== K-FOLD RESULT : Linear Regression =====
Mean R2   : 0.840414
Mean MAE  : 0.183352
Mean RMSE : 0.236638
Mean MAPE : 0.044305
```

```
•••
===== Polynomial Linear Regression =====
R2   : 0.851887
MAE  : 0.174527
RMSE : 0.219238
MAPE : 0.040811
```

```
•••
===== K-FOLD RESULT : Polynomial Linear Regression =====
Mean R2   : 0.846139
Mean MAE  : 0.179897
Mean RMSE : 0.232174
Mean MAPE : 0.043048
```

b. Mengunakan Mean

```
evaluate_model(y_test, y_pred_baseline_LR, "Baseline Linear Regression")
evaluate_model(y_test, y_pred_baseline_DT, "Baseline Decision Tree")
```

```
•••
===== Baseline Linear Regression =====
R2   : 0.836402
MAE  : 0.184987
RMSE : 0.230414
MAPE : 0.043594

===== Baseline Decision Tree =====
R2   : 0.698919
MAE  : 0.240462
RMSE : 0.312580
MAPE : 0.055929
```

```
•••
===== Polynomial Linear Regression =====
R2   : 0.852135
MAE  : 0.174570
RMSE : 0.219055
MAPE : 0.040810
```

```
•••
===== K-FOLD RESULT : Linear Regression =====
Mean R2   : 0.840088
Mean MAE  : 0.183548
Mean RMSE : 0.236885
Mean MAPE : 0.044354
```

```
===== K-FOLD RESULT : Polynomial Linear Regression =====
Mean R2   : 0.845728
Mean MAE  : 0.179989
Mean RMSE : 0.232490
Mean MAPE : 0.043079
```

4. Experiment Alpha

```
import numpy as np

param_grid = {
    "alpha": [0.001, 0.01, 0.1, 1, 10, 50, 100]
}

param_grid_lasso = {
    "alpha": [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1, 1]
}

rmse_scorer = make_scorer(
    mean_squared_error,
    greater_is_better=False,
    squared=False
)
```

a. **Ridge regression**

```
print("Best Alpha:", grid_search.best_params_)
```

```
Best Alpha: {'alpha': 0.001}
```

```
best_ridge = grid_search.best_estimator_

pred_log = best_ridge.predict(X_test_scaled_df)
pred = np.expm1(pred_log)

evaluate_model(y_test, pred, "Best Ridge (GridSearch)")
```

```
===== Best Ridge (GridSearch) =====
R2   : 0.818818
MAE  : 0.192531
RMSE : 0.242481
MAPE : 0.045285
```

b. **Lasso Regression**

```
print("Best Alpha:", grid_search_lasso.best_params_)
```

```
Best Alpha: {'alpha': 0.0001}
```

```
best_lasso = grid_search_lasso.best_estimator_

pred_log = best_lasso.predict(X_test_scaled_df)
pred = np.expm1(pred_log)

evaluate_model(y_test, pred, "Best Lasso (GridSearch)")
```

```
===== Best Lasso (GridSearch) =====
R2   : 0.819006
MAE  : 0.192571
RMSE : 0.242355
MAPE : 0.045305
```