best\_params = report\_perf(opt, X\_train, y\_train,'XGBoost\_regression',

callbacks=[overdone\_control, time\_limit\_control])

XGBoost\_regression took 11287.00 seconds, candidates checked: 62, best CV score: -0.709 ± 0.000

Best parameters:

OrderedDict([('colsample\_bytree', 0.13641457979856397),

('learning\_rate', 0.09604083138419779),

('max\_depth', 5),

('n\_estimators', 2206),

('reg\_alpha', 68.85614181114505),

('reg\_lambda', 9.260067417192285),

('subsample', 0.8980218163372579)])

Prediction on test data

Having got the best hyperparameters for the data at hand, we instantiate a XGBoost using such values and train our model on all the available examples.

After having trained the model, we predict on the test set and we save the results on a csv file.

In [19]:

*# Transferring the best parameters to our basic regressor*

reg = XGBRegressor(random\_state=0, booster='gbtree', objective='reg:squarederror', tree\_method='gpu\_hist', \*\*best\_params)

In [20]:

*# Cross-validation prediction*

folds = 10

skf = StratifiedKFold(n\_splits=folds,

shuffle=True,

random\_state=0)

predictions = np.zeros(len(X\_test))

rmse = list()

for k, (train\_idx, val\_idx) **in** enumerate(skf.split(X\_train, y\_stratified)):

reg.fit(X\_train.iloc[train\_idx, :], y\_train[train\_idx])

val\_preds = reg.predict(X\_train.iloc[val\_idx, :])

val\_rmse = mean\_squared\_error(y\_true=y\_train[val\_idx], y\_pred=val\_preds, squared=False)

print(f"Fold **{**k**}** RMSE: **{**val\_rmse**:**0.5f**}**")

rmse.append(val\_rmse)

predictions += reg.predict(X\_test).ravel()

predictions /= folds

print(f"repeated CV RMSE: **{**np.mean(rmse)**:**0.5f**}** (std=**{**np.std(rmse)**:**0.5f**}**)")

Fold 0 RMSE: 0.70777

Fold 1 RMSE: 0.71033

Fold 2 RMSE: 0.70861

Fold 3 RMSE: 0.70917

Fold 4 RMSE: 0.70706

Fold 5 RMSE: 0.70931

Fold 6 RMSE: 0.71208

Fold 7 RMSE: 0.70771

Fold 8 RMSE: 0.70977

Fold 9 RMSE: 0.70757

repeated CV RMSE: 0.70894 (std=0.00145)

In [21]:

*# Preparing the submission*

submission = pd.DataFrame({'id':X\_test.index,

'target': predictions})

submission.to\_csv("submission.csv", index = False)

In [22]:

submission

Out[22]:

|  | id | target |
| --- | --- | --- |
| 0 | 0 | 8.145761 |
| 1 | 5 | 8.392107 |
| 2 | 15 | 8.405922 |
| 3 | 16 | 8.532095 |
| 4 | 17 | 8.160963 |
| ... | ... | ... |
| 199995 | 499987 | 8.027081 |
| 199996 | 499990 | 8.451505 |
| 199997 | 499991 | 8.457389 |
| 199998 | 499994 | 8.156412 |
| 199999 | 499995 | 7.992031 |

200000 rows × 2 columns

In [ ]: