Validación métricas con HiperParametros - Laboratorio 3

Johan Steven Benavides Guarnizo-88593

Sebastian Morales Devia - 73487

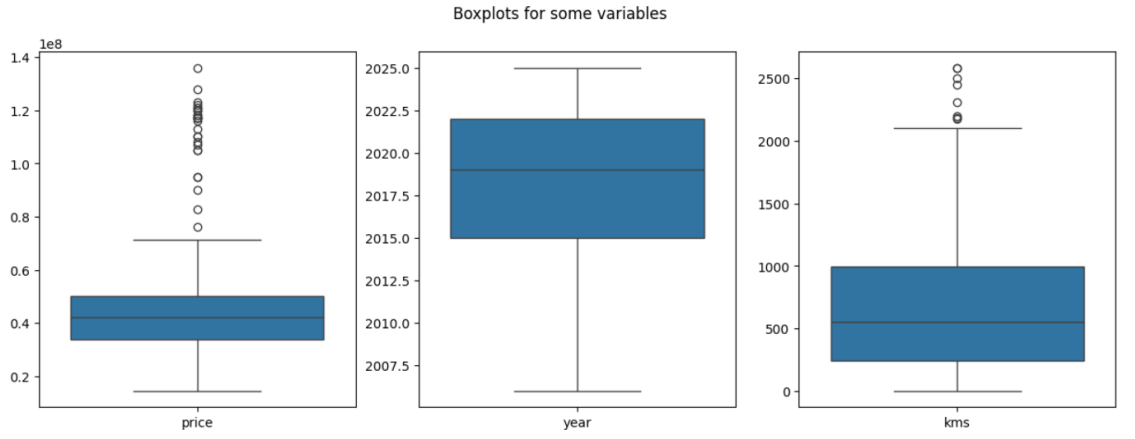
Universidad ECCI
Facultad de Ingeniería

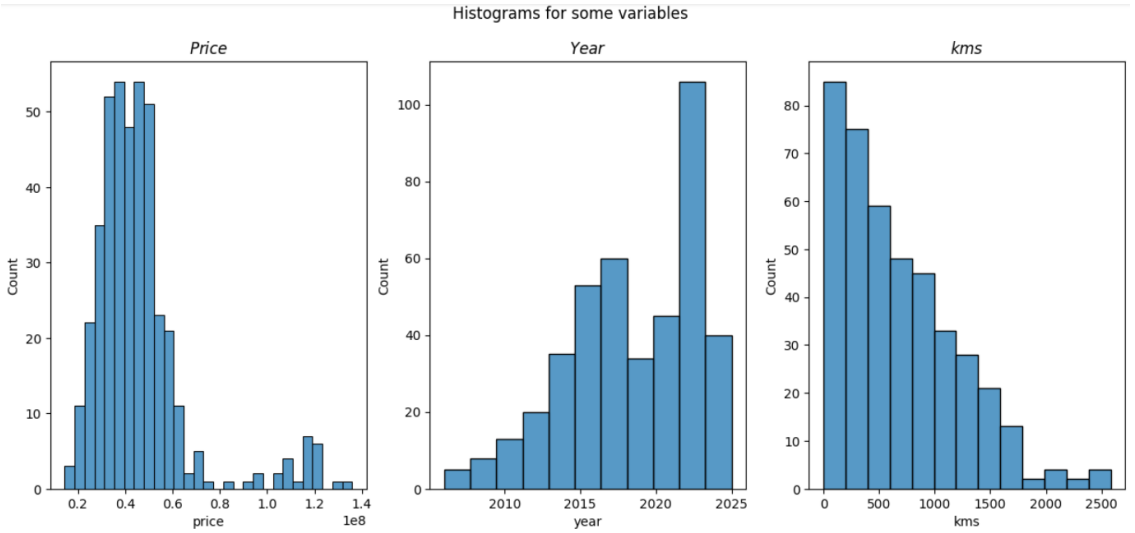
Elias Buitrago Bolivar

Bogota
2024

Resultados de los diferentes modelos con los datos subidos sin seleccionar ni filtrar ninguna info:

Graficas:





Multivariate lineal regression

Multivariate lineal regression

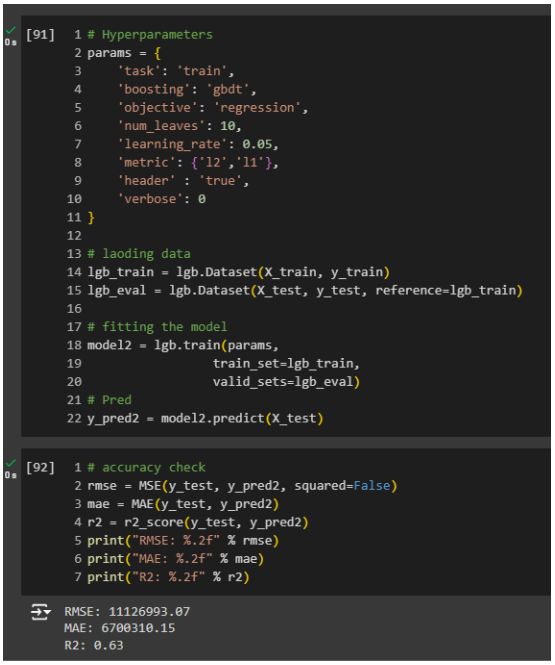
```
[87]  1 # Define model and prediction
      2 ols = LinearRegression()
      3 model1 = ols.fit(X_train, y_train)
      4 y_pred1 = model1.predict(X_test)
```

```
[88]  1 # accuracy check
      2 rmse = MSE(y_test, y_pred1, squared=False)
      3 mae = MAE(y_test, y_pred1)
      4 r2 = r2_score(y_test, y_pred1)
      5 print("RMSE: %.2f" % rmse)
      6 print("MAE: %.2f" % mae)
      7 print("R2: %.2f" % r2)
```

```
RMSE: 12109549.03
MAE: 8817707.26
R2: 0.57
```

RMSE: 12109549.03
MAE: 8817707.26
R2: 0.57

Light GBM

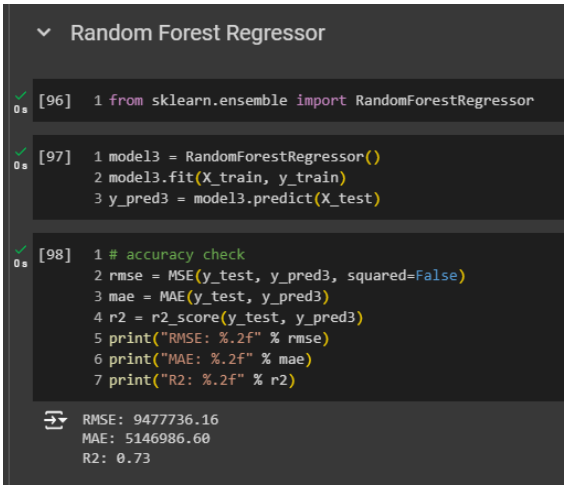


```
[91]  1 # Hyperparameters
      2 params = {
      3     'task': 'train',
      4     'boosting': 'gbdt',
      5     'objective': 'regression',
      6     'num_leaves': 10,
      7     'learning_rate': 0.05,
      8     'metric': {'l2','l1'},
      9     'header' : 'true',
     10     'verbose': 0
     11 }
     12
     13 # laoding data
     14 lgb_train = lgb.Dataset(X_train, y_train)
     15 lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)
     16
     17 # fitting the model
     18 model2 = lgb.train(params,
     19                    train_set=lgb_train,
     20                    valid_sets=lgb_eval)
     21 # Pred
     22 y_pred2 = model2.predict(X_test)
```

```
[92]  1 # accuracy check
      2 rmse = MSE(y_test, y_pred2, squared=False)
      3 mae = MAE(y_test, y_pred2)
      4 r2 = r2_score(y_test, y_pred2)
      5 print("RMSE: %.2f" % rmse)
      6 print("MAE: %.2f" % mae)
      7 print("R2: %.2f" % r2)

RMSE: 11126993.07
MAE: 6700310.15
R2: 0.63
```

RMSE: 11126993.07
MAE: 6700310.15
R2: 0.63

Random Forest Regressor



Random Forest Regressor

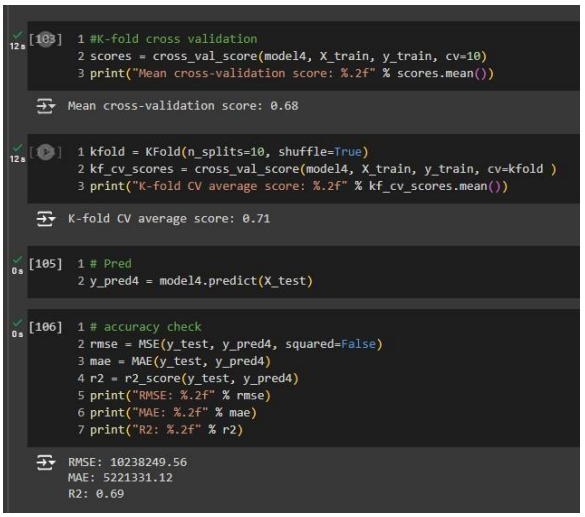
```
[96]  1 from sklearn.ensemble import RandomForestRegressor
```

```
[97]  1 model3 = RandomForestRegressor()
      2 model3.fit(X_train, y_train)
      3 y_pred3 = model3.predict(X_test)
```

```
[98]  1 # accuracy check
      2 rmse = MSE(y_test, y_pred3, squared=False)
      3 mae = MAE(y_test, y_pred3)
      4 r2 = r2_score(y_test, y_pred3)
      5 print("RMSE: %.2f" % rmse)
      6 print("MAE: %.2f" % mae)
      7 print("R2: %.2f" % r2)

RMSE: 9477736.16
MAE: 5146986.60
R2: 0.73
```

RMSE: 9477736.16
MAE: 5146986.60
R2: 0.73

Xgboost regressor

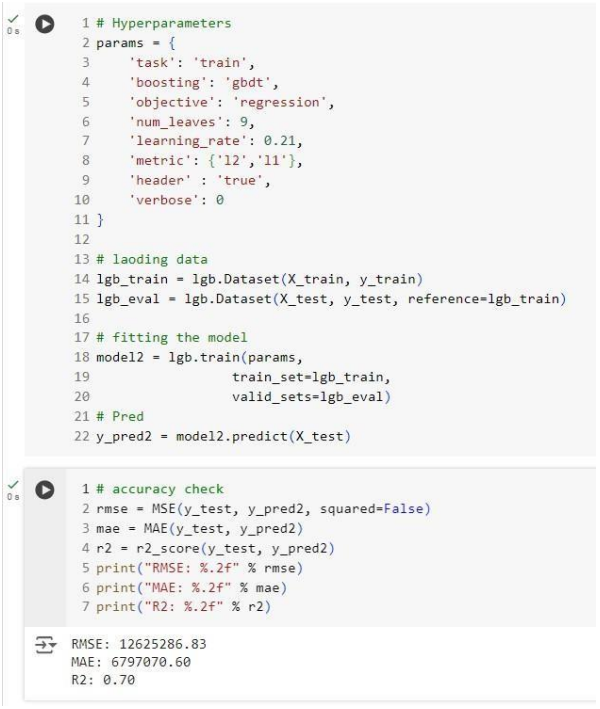


```
[103] 1 #K-fold cross validation
      2 scores = cross_val_score(model4, X_train, y_train, cv=10)
      3 print("Mean cross-validation score: %.2f" % scores.mean())

      Mean cross-validation score: 0.68

      1 kfold = KFold(n_splits=10, shuffle=True)
      2 kf_cv_scores = cross_val_score(model4, X_train, y_train, cv=kfold )
      3 print("K-fold CV average score: %.2f" % kf_cv_scores.mean())

      K-fold CV average score: 0.71

[105] 1 # Pred
      2 y_pred4 = model4.predict(X_test)

[106] 1 # accuracy check
      2 rmse = MSE(y_test, y_pred4, squared=False)
      3 mae = MAE(y_test, y_pred4)
      4 r2 = r2_score(y_test, y_pred4)
      5 print("RMSE: %.2f" % rmse)
      6 print("MAE: %.2f" % mae)
      7 print("R2: %.2f" % r2)

      RMSE: 10238249.56
      MAE: 5221331.12
      R2: 0.69
```

RMSE: 10238249.56
MAE: 5221331.12
R2: 0.69

Prueba 1:

Light GBM



```
 1 # Hyperparameters
 2 params = {
 3     'task': 'train',
 4     'boosting': 'gbdt',
 5     'objective': 'regression',
 6     'num_leaves': 9,
 7     'learning_rate': 0.21,
 8     'metric': {'l2','l1'},
 9     'header' : 'true',
10     'verbose': 0
11 }
12
13 # laoding data
14 lgb_train = lgb.Dataset(X_train, y_train)
15 lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)
16
17 # fitting the model
18 model2 = lgb.train(params,
19                    train_set=lgb_train,
20                    valid_sets=lgb_eval)
21 # Pred
22 y_pred2 = model2.predict(X_test)
```

```
 1 # accuracy check
 2 rmse = MSE(y_test, y_pred2, squared=False)
 3 mae = MAE(y_test, y_pred2)
 4 r2 = r2_score(y_test, y_pred2)
 5 print("RMSE: %.2f" % rmse)
 6 print("MAE: %.2f" % mae)
 7 print("R2: %.2f" % r2)

RMSE: 12625286.83
MAE: 6797070.60
R2: 0.70
```

RMSE: 12625286.83
MAE: 6797070.60
R2: 0.70

Random Forest Regressor

```
[ ]  1 from sklearn.ensemble import RandomForestRegressor
```

```
[ ]  1 model3 = RandomForestRegressor()
     2 model3.fit(X_train, y_train)
     3 y_pred3 = model3.predict(X_test)
```

```
[ ]  1 # accuracy check
     2 rmse = MSE(y_test, y_pred3, squared=False)
     3 mae = MAE(y_test, y_pred3)
     4 r2 = r2_score(y_test, y_pred3)
     5 print("RMSE: %.2f" % rmse)
     6 print("MAE: %.2f" % mae)
     7 print("R2: %.2f" % r2)
```

```
RMSE: 8993766.98
MAE: 5157716.40
R2: 0.85
```

RMSE: 8993766.98
MAE: 5157716.40
R2: 0.85

Xgboost regressor

```
[ ]  1 # Pred
     2 y_pred4 = model4.predict(X_test)
```

```
[ ]  1 # accuracy check
     2 rmse = MSE(y_test, y_pred4, squared=False)
     3 mae = MAE(y_test, y_pred4)
     4 r2 = r2_score(y_test, y_pred4)
     5 print("RMSE: %.2f" % rmse)
     6 print("MAE: %.2f" % mae)
     7 print("R2: %.2f" % r2)
```

```
RMSE: 11535189.22
MAE: 6249334.76
R2: 0.75
```

RMSE: 11535189.22
MAE: 6249334.76
R2: 0.75

Prueba 2:

Light GBM

```
[80]  # Hyperparameters
      params = {
          'task': 'train',
          'boosting': 'gbdt',
          'objective': 'regression',
          'num_leaves': 5,
          'learning_rate': 0.52,
          'metric': {'l2','l1'},
          'header' : 'true',
          'verbose': 0
      }

      # laoding data
      lgb_train = lgb.Dataset(X_train, y_train)
      lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)

      # fitting the model
      model2 = lgb.train(params,
                         train_set=lgb_train,
                         valid_sets=lgb_eval)
      # Pred
      y_pred2 = model2.predict(X_test)
```

```
# accuracy check
rmse = MSE(y_test, y_pred2, squared=False)
mae = MAE(y_test, y_pred2)
r2 = r2_score(y_test, y_pred2)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 11861025.88
MAE: 6673291.02
R2: 0.74
```

RMSE: 11861025.88
MAE: 6673291.02
R2: 0.74

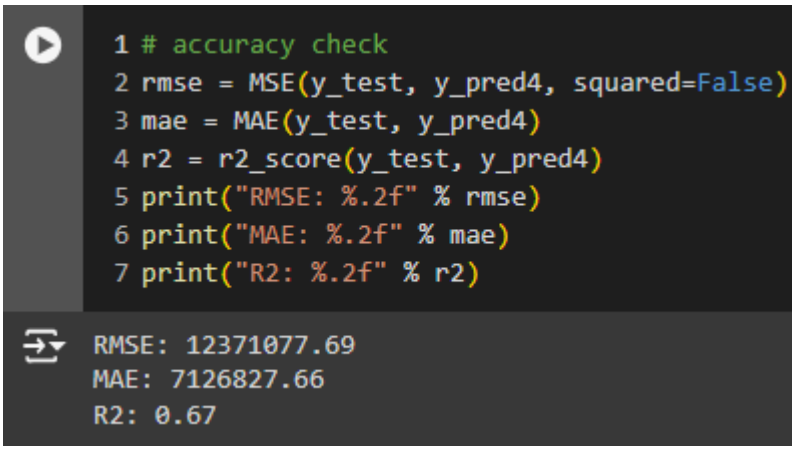
Random Forest Regressor

```
param_grid = {
    'n_estimators': [50, 100, 200, 300],  # Número de árboles en
    'max_depth': [None, 10, 20, 30, 40, 50],  # Profundidad máxi
    'min_samples_split': [2, 5, 10],  # Número mínimo de muestra
    'min_samples_leaf': [1, 2, 4],  # Número mínimo de muestras
    'max_features': ['auto', 'sqrt', 'log2']  # Número máximo de
}
rf = RandomForestRegressor(random_state=42)
random_search = RandomizedSearchCV(estimator=rf, param_distribut
                                   scoring=scoring, refit='rmse'


random_search.fit(X_train, y_train)
best_model = random_search.best_estimator_
y_pred = best_model.predict(X_test)


rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
RMSE: 8900388.53
MAE: 5350869.41
R2: 0.83
```

RMSE: 8900388.53
MAE: 5350869.41
R2: 0.83

Xgboost regressor



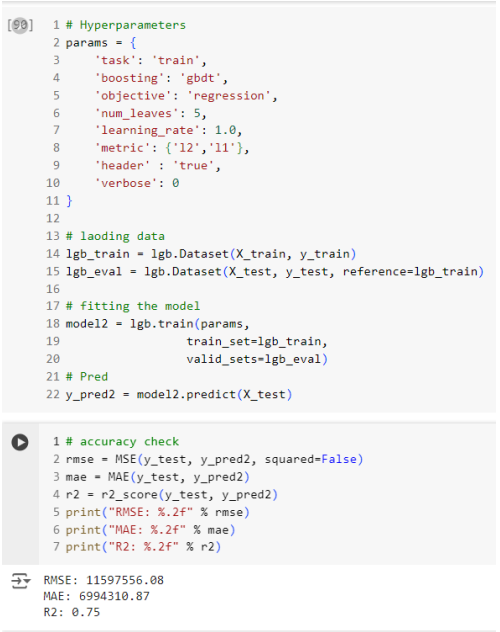
```
# accuracy check
rmse = MSE(y_test, y_pred4, squared=False)
mae = MAE(y_test, y_pred4)
r2 = r2_score(y_test, y_pred4)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 12371077.69
MAE: 7126827.66
R2: 0.67
```

RMSE: 12371077.69
MAE: 7126827.66
R2: 0.67

Prueba 3:

Light GBM



```
# Hyperparameters
params = {
    'task': 'train',
    'boosting': 'gbdt',
    'objective': 'regression',
    'num_leaves': 5,
    'learning_rate': 1.0,
    'metric': {'l2','l1'},
    'header' : 'true',
    'verbose': 0
}

# laoding data
lgb_train = lgb.Dataset(X_train, y_train)
lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)

# fitting the model
model2 = lgb.train(params,
                   train_set=lgb_train,
                   valid_sets=lgb_eval)
# Pred
y_pred2 = model2.predict(X_test)
```

```
# accuracy check
rmse = MSE(y_test, y_pred2, squared=False)
mae = MAE(y_test, y_pred2)
r2 = r2_score(y_test, y_pred2)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 11597556.08
MAE: 6994310.87
R2: 0.75
```

RMSE: 11597556.08
MAE: 6994310.87
R2: 0.75

Random Forest Regressor

```
param_grid = {
    'n_estimators': [ 150, 200, 350],  # Número de árboles en el
    'max_depth': [None, 10, 20],  # Profundidad máxima de los ár
    'min_samples_split': [2, 9, 10],  # Número mínimo de muestra
    'min_samples_leaf': [2, 3, 4],  # Número mínimo de muestras
    'max_features': ['auto', 'sqrt', 'log2']  # Número máximo de
}
rf = RandomForestRegressor(random_state=42)
random_search = RandomizedSearchCV(estimator=rf, param_distribut
                                   scoring=scoring, refit='rmse'


random_search.fit(X_train, y_train)
best_model = random_search.best_estimator_
y_pred = best_model.predict(X_test)


rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
RMSE: 8684526.04
MAE: 5170397.83
R2: 0.84
```

RMSE: 8684526.04
MAE: 5170397.83
R2: 0.84

Xgboost regressor

```
[50]  1 # Pred
      2 y_pred4 = model4.predict(X_test)
```

```
1 # accuracy check
2 rmse = MSE(y_test, y_pred4, squared=False)
3 mae = MAE(y_test, y_pred4)
4 r2 = r2_score(y_test, y_pred4)
5 print("RMSE: %.2f" % rmse)
6 print("MAE: %.2f" % mae)
7 print("R2: %.2f" % r2)
```

```
RMSE: 11715237.49
MAE: 6850894.90
R2: 0.71
```

RMSE: 11715237.49
MAE: 6850894.90
R2: 0.71

Prueba 4:

Light GBM

```
12
13 # laoding data
14 lgb_train = lgb.Dataset(X_train, y_train)
15 lgb_eval = lgb.Dataset(X_test, y_test, reference=lgb_train)
16
17 # fitting the model
18 model2 = lgb.train(params,
19                     train_set=lgb_train,
20                     valid_sets=lgb_eval)
21 # Pred
22 y_pred2 = model2.predict(X_test)
```

```
1 # accuracy check
2 rmse = MSE(y_test, y_pred2, squared=False)
3 mae = MAE(y_test, y_pred2)
4 r2 = r2_score(y_test, y_pred2)
5 print("RMSE: %.2f" % rmse)
6 print("MAE: %.2f" % mae)
7 print("R2: %.2f" % r2)
```

```
RMSE: 11228274.90
MAE: 6582336.57
R2: 0.76
```

RMSE: 11228274.90
MAE: 6582336.57
R2: 0.76

Random Forest Regressor

```
param_grid = {
    'n_estimators': [ 200, 400],  # Número de árboles en el bo
    'max_depth': [None, 10, 20, 30],  # Profundidad máxima de
    'min_samples_split': [2, 9, 10],  # Número mínimo de muest
    'min_samples_leaf': [2, 3, 4, 5],  # Número mínimo de mues
    'max_features': ['auto', 'sqrt', 'log2']  # Número máximo
}
rf = RandomForestRegressor(random_state=42)
random_search = RandomizedSearchCV(estimator=rf, param_distrib
                                   scoring=scoring, refit='rms


random_search.fit(X_train, y_train)
best_model = random_search.best_estimator_
y_pred = best_model.predict(X_test)


rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
RMSE: 8955562.17
MAE: 5367564.00
R2: 0.83
```

RMSE: 8955562.17
MAE: 5367564.00
R2: 0.83

Xgboost regressor

```
[63]  # Pred
      y_pred4 = model4.predict(X_test)
```

```
# accuracy check
rmse = MSE(y_test, y_pred4, squared=False)
mae = MAE(y_test, y_pred4)
r2 = r2_score(y_test, y_pred4)
print("RMSE: %.2f" % rmse)
print("MAE: %.2f" % mae)
print("R2: %.2f" % r2)
```

```
RMSE: 10775624.04
MAE: 6505791.87
R2: 0.75
```

RMSE: 10775624.04
MAE: 6505791.87
R2: 0.75