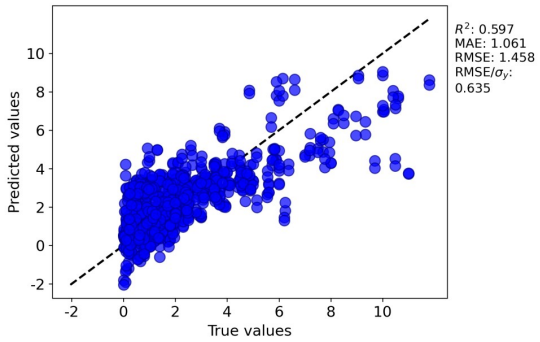
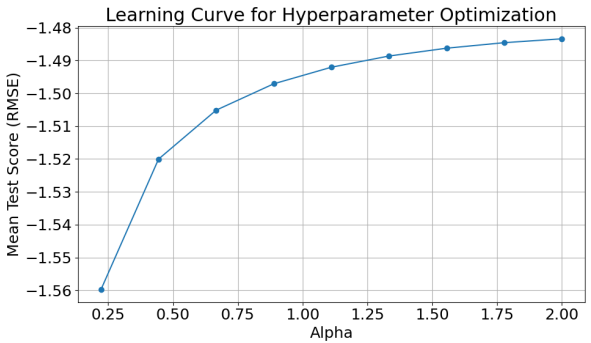
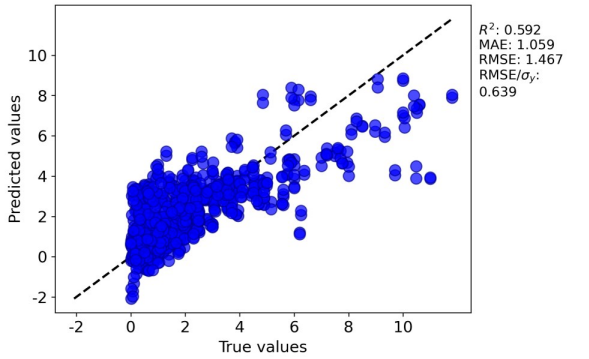
Assessment Figures







ML4ER Assignment 5

Jiahui Yang
Informatics Skunkworks
MSE 401, 3 Credits
Aug 01 2024

Progress

- Section 5 Step 1 and Step 2

```
[128]
# Step 1
from sklearn.linear_model import Ridge
✓ 0.0s
```

```
[129]
# Step 2: Build the default model
default_ridge = SklearnModel(model='Ridge')
models = [default_ridge]
metrics = ['r2_score', 'mean_absolute_error', 'root_mean_squared_error', 'rmse_over_stdev']
✓ 0.0s
```

Import the Ridge model from sklearn.linear_model and create a default Ridge model using SklearnModel, specifying the evaluation metrics.

Progress

- Section 5 Step 2

Evaluate performance on the test data using the NoSplit class, then perform 5-fold cross-validation using RepeatedKFold with 2 repeats.

```
# Step 2: Evaluate performance on the test data using NoSplit class
splitter = NoSplit()
splitter.evaluate(X=X,
                  y=y,
                  models=models,
                  preprocessor=None,
                  metrics=metrics,
                  savepath=savepath,
                  X_extra=X_extra,
                  leaveout_inds=X_testdata,
                  verbosity=3)
```

✓ 4.9s

```
# Step 2: Perform 5-fold cross-validation
splitter = SklearnDataSplitter(splitter='RepeatedKFold', n_repeats=2, n_splits=5)
splitter.evaluate(X=X,
                  y=y,
                  models=models,
                  preprocessor=None,
                  selectors=[NoSelect()],
                  metrics=metrics,
                  savepath=savepath,
                  X_extra=X_extra,
                  leaveout_inds=X_testdata,
                  recalibrate_errors=True,
                  verbosity=3)
```

✓ 11.0s

Progress

- Section 5 Step 3

Perform a grid search on the alpha hyperparameter using NoSplit, then conduct 5-fold cross-validation with the optimized model using RepeatedKFold (2 repeats, 5 splits).

```
# Step 3: Perform Grid Search on the Alpha Hyperparameter
grid_ridge = SklearnModel(model='Ridge')
models = [grid_ridge]
grid1 = GridSearch(param_names='alpha', param_values='0 2 10 lin float', scoring='root_mean_squared_error')
grids = [grid1]
splitter = NoSplit()
splitter.evaluate(X=X,
                  y=y,
                  models=models,
                  preprocessor=None,
                  metrics=metrics,
                  savepath=savepath,
                  X_extra=X_extra,
                  leaveout_inds=X_testdata,
                  hyperopts=grids,
                  recalibrate_errors=True,
                  verbosity=3)
```

[32] ✓ 4.7s

```
# Step 3: Perform 5-Fold Cross-Validation with the Optimized Model
optimized_ridge = SklearnModel(model='Ridge', alpha=2)  # Replace alpha with the optimal value found from grid search
models = [optimized_ridge]
splitter = SklearnDataSplitter(splitter='RepeatedKFold', n_repeats=2, n_splits=5)
splitter.evaluate(X=X,
                  y=y,
                  models=models,
                  preprocessor=None,
                  selectors=[NoSelect()],
                  metrics=metrics,
                  savepath=savepath,
                  X_extra=X_extra,
                  leaveout_inds=X_testdata,
                  recalibrate_errors=True,
                  verbosity=3)
```

[33] ✓ 12.4s

	A	B	C
1		alpha	
2	Best Parameters	2	
3			

Progress

- Section 5 Step 3 Plot

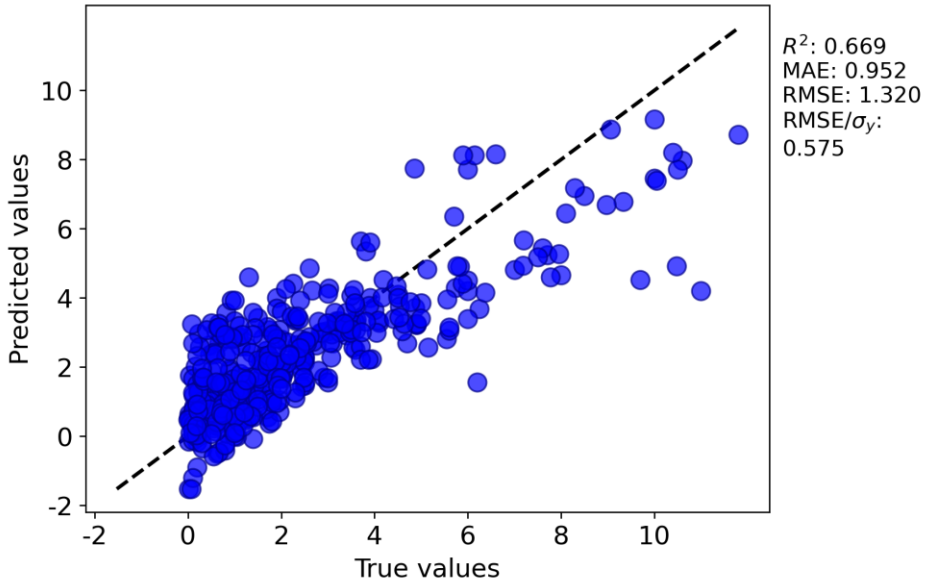
Python code that reads GridSearch results from an Excel file and plots a learning curve of alpha values against mean_test_score (RMSE).

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the results from the GridSearch output file
grid_search_results = pd.read_excel('./Nanohub_workflow/Ridge_NoSplit_NoPreprocessor_NoSelect_2024_07_30_13_09_50/split_out

# Plotting the learning curve
plt.figure(figsize=(10, 6))
plt.plot(grid_search_results['alpha'], grid_search_results['mean_test_score'], marker='o')
plt.xlabel('Alpha')
plt.ylabel('Mean Test Score (RMSE)')
plt.title('Learning Curve for Hyperparameter Optimization')
plt.grid(True)
plt.show()
```

[11] ✓ 0.1s

Progress

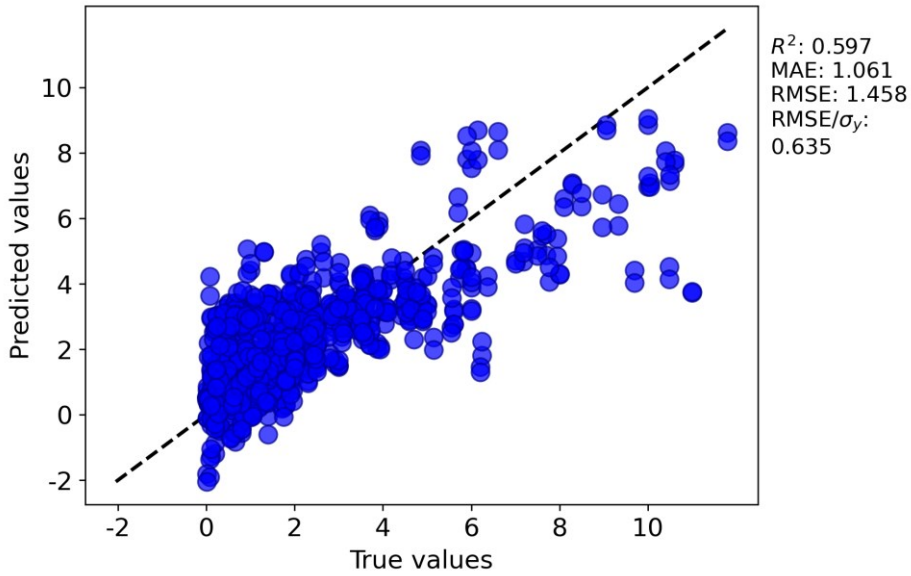
- Test data parity plot of your default chosen model from Section 5 (NoSplit)



This scatter plot shows predicted values versus true values, with an R^2 of 0.669, MAE of 0.952, RMSE of 1.320, and RMSE/σ_Y of 0.575.

Progress

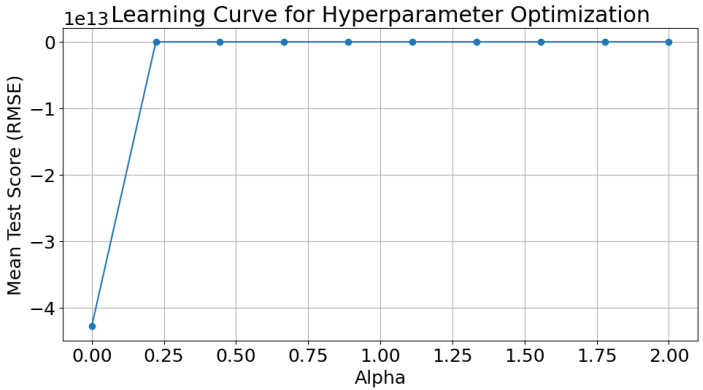
- Test data parity plot of your default chosen model from Section 5 (5-fold cross-validation)

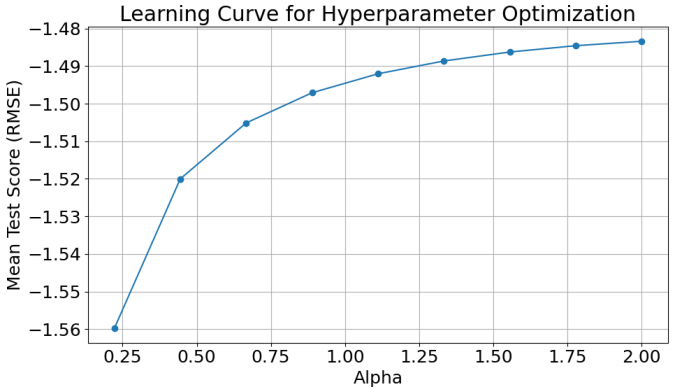


This scatter plot shows predicted values versus true values, with an R^2 of 0.597, MAE of 1.061, RMSE of 1.458, and RMSE/σ_y of 0.635.

Progress

- Learning curve for your hyperparameter optimization from Section 5 (NoSplit)
- /split_outer_0/Split_0/*_Output.xlsx

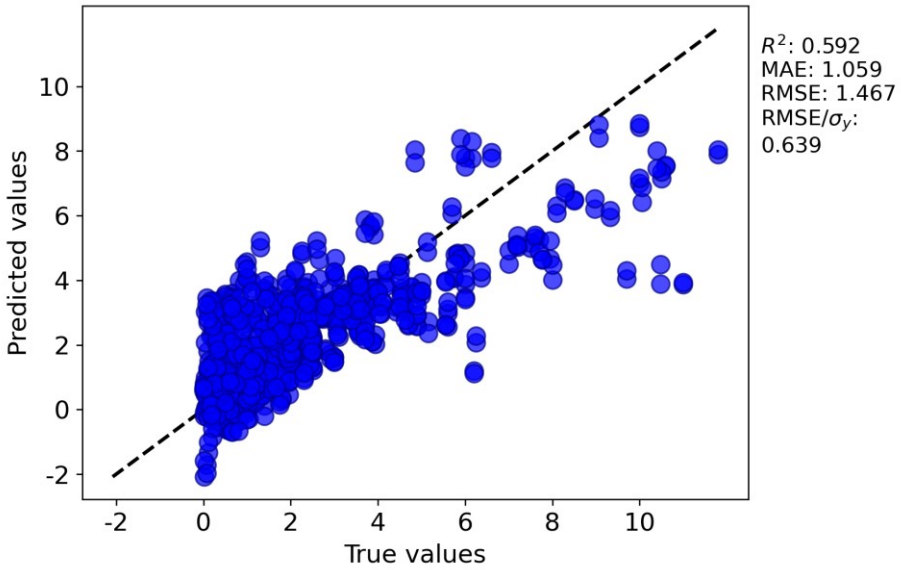




As alpha increases from 0.25 to 2.0, the RMSE value decreases, indicating improved model performance. The curve highlights how different alpha values affect the model's prediction accuracy.

Progress

- Test data parity plot of your optimized model performance from Section 5 (5-fold cross-validation)



This scatter plot shows predicted values versus true values, with an R^2 of 0.592, MAE of 1.059, RMSE of 1.467, and RMSE/σ_Y of 0.639.

Problems

- I had an issue caused by the version of the MAST-ML installation. Changing featurize_df to composition_df resolved the problem.

```
generator = ElementalFeatureGenerator(featurize_df = X_extra["chemicalFormula Clean"],
                                      feature_types='composition_avg',
                                      remove_constant_columns=True)
X, y = generator.evaluate(X = X,
                          y = y,
                          savepath = savepath)
```

[31] ⊗ 0.0s

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[31], line 1
----> 1 generator = ElementalFeatureGenerator(featurize_df = X_extra["chemicalFormula Clean"],
      2                                       feature_types='composition_avg',
      3                                       remove_constant_columns=True)
      4 X, y = generator.evaluate(X = X,
      5                           y = y,
      6                           savepath = savepath)

TypeError: ElementalFeatureGenerator.__init__() got an unexpected keyword argument 'featurize_df'
```

Questions

- Can I use GridSearch along with cross-validation simultaneously?

```
# Step 3: Perform Grid Search on the Alpha Hyperparameter
grid_ridge = SklearnModel(model='Ridge')
models = [grid_ridge]
grid1 = GridSearch(param_names='alpha', param_values='0 2 10 lin float', scoring='root_mean_squared_error')
grids = [grid1]
splitter = SklearnDataSplitter(splitter='RepeatedKFold', n_repeats=2, n_splits=5)
splitter.evaluate(X=X,
                  y=y,
                  models=models,
                  preprocessor=None,
                  selectors=[NoSelect()],
                  metrics=metrics,
                  savepath=savepath,
                  X_extra=X_extra,
                  leaveout_inds=X_testdata,
                  recalibrate_errors=True,
                  verbosity=3)
```

Hours Summary

Date	Hours	Description of Work
07/30/2024	5 hours	Complete module 4: Comparing Model types