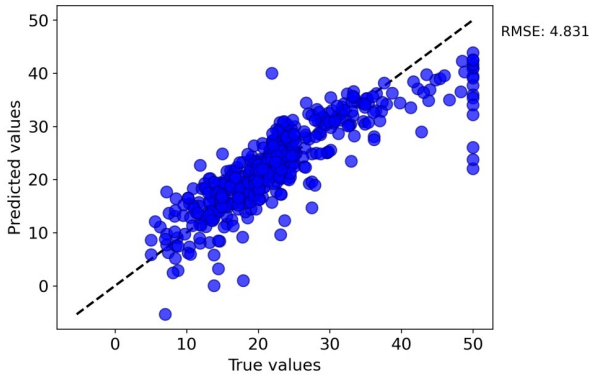
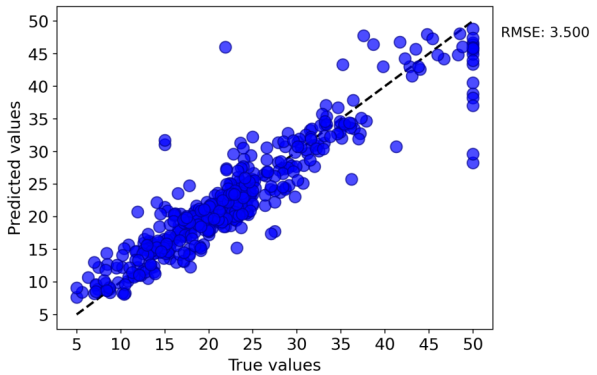
Assessment Figures

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	-0.419782	0.284830	-1.287909	-0.272599	-0.144217	0.413672	-0.120013	0.140214	-0.982843	-0.666608	-1.459000	0.441052	-1.075562
1	-0.417339	-0.487722	-0.593381	-0.272599	-0.740262	0.194274	0.367166	0.557160	-0.867883	-0.987329	-0.303094	0.441052	-0.492439
2	-0.417342	-0.487722	-0.593381	-0.272599	-0.740262	1.282714	-0.265812	0.557160	-0.867883	-0.987329	-0.303094	0.396427	-1.208727
3	-0.416750	-0.487722	-1.306878	-0.272599	-0.835284	1.016303	-0.809889	1.077737	-0.752922	-1.106115	0.113032	0.416163	-1.361517
4	-0.412482	-0.487722	-1.306878	-0.272599	-0.835284	1.228577	-0.511180	1.077737	-0.752922	-1.106115	0.113032	0.441052	-1.026501





ML4ER Assignment 4

Jiahui Yang
Informatics Skunkworks
MSE 401, 3 Credits
Jul 31 2024

Progress

- Normalized features in “task 3”

The normalized data has a mean of 0 and a variance of 1, confirming that the normalization process was successful.

```
preprocessor = SklearnPreprocessor(
                                   preprocessor='StandardScaler', as_frame=True
                                  )
```
[14] ✓ 0.0s

```
X = preprocessor.evaluate(df_X, savepath=savepath)
```
[16] ✓ 0.0s

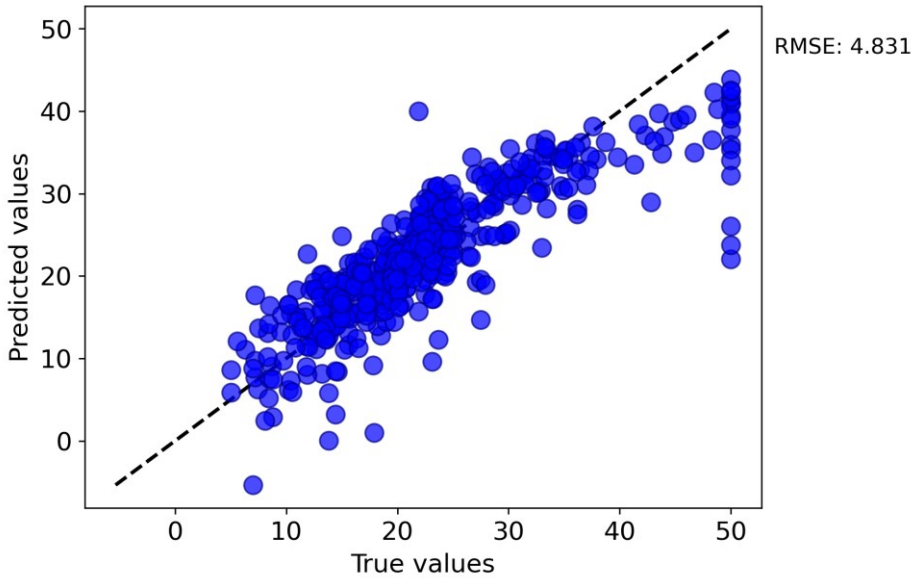
We can then verify that the normalization took place as expected by outputting the X dataframe

```
X.head(5)
```
[17] ✓ 0.0s

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	-0.419782	0.284830	-1.287909	-0.272599	-0.144217	0.413672	-0.120013	0.140214	-0.982843	-0.666608	-1.459000	0.441052	-1.075562
1	-0.417339	-0.487722	-0.593381	-0.272599	-0.740262	0.194274	0.367166	0.557160	-0.867883	-0.987329	-0.303094	0.441052	-0.492439
2	-0.417342	-0.487722	-0.593381	-0.272599	-0.740262	1.282714	-0.265812	0.557160	-0.867883	-0.987329	-0.303094	0.396427	-1.208727
3	-0.416750	-0.487722	-1.306878	-0.272599	-0.835284	1.016303	-0.809889	1.077737	-0.752922	-1.106115	0.113032	0.416163	-1.361517
4	-0.412482	-0.487722	-1.306878	-0.272599	-0.835284	1.228577	-0.511180	1.077737	-0.752922	-1.106115	0.113032	0.441052	-1.026501

Progress

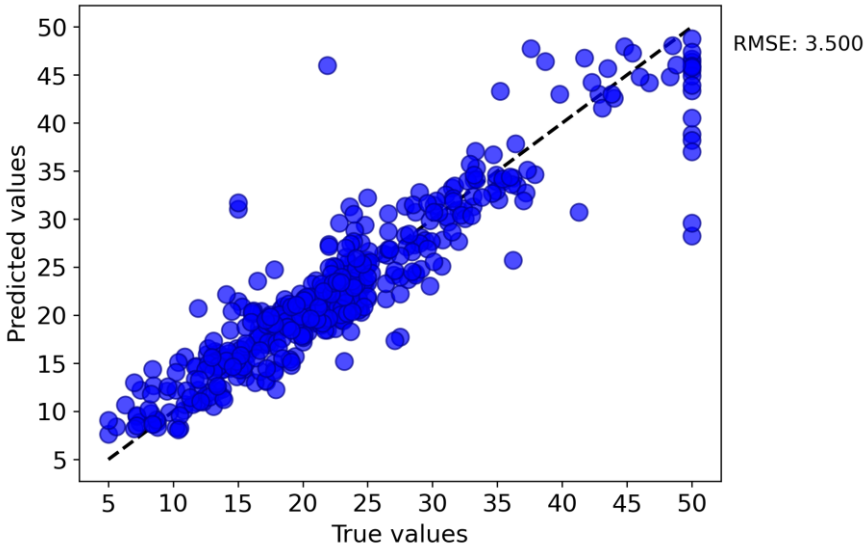
- 5-fold CV parity plot of the linear regression model from “task 5”



The scatter plot compares true values (x-axis) and predicted values (y-axis) of a machine learning model. Each blue dot represents a data point, with the dashed line indicating ideal predictions. The RMSE is 4.831.

Progress

- 5-fold CV parity plot of the random forest regressor model from “task 6”



The scatter plot compares true values (x-axis) and predicted values (y-axis) of a machine learning model. Each blue dot represents a data point, with the dashed line indicating ideal predictions. The RMSE is 3.500.

Problems

- I need to use another way to import data.

```
SklearnDatasets
```
[7] ✓ 0.0s

```
mastml.datasets.SklearnDatasets
```

```
X, y = SklearnDatasets(as_frame=True).load_housing()
```
[15] ⊗ 0.0s

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[15], line 1
----> 1 X, y = SklearnDatasets(as_frame=True).load_housing()

AttributeError: 'SklearnDatasets' object has no attribute 'load_housing'
```

```
X = boston.data
y = boston.target
```
[12] ✓ 0.0s

Problems

- The data needs to be converted into a DataFrame for ML training.

We can output both of these objects to check that they have imported correctly. We should expect to see two Pandas DataFrames that both have the same number of rows, that the features dataframe has the appropriate number of columns (13), and that the correct column has been assigned to the target dataframe (MEDV). We know from the previous dataset description output that the MEDV is the median values of homes.

```
11]  #X
     ✓ 0.0s                                                    Python
```

```
12]  #y
     ✓ 0.0s                                                    Python
```

```
13]  import pandas as pd

     df_X = pd.DataFrame(X, columns=boston.feature_names)
     df_y = pd.DataFrame(y, columns=["MEDV"])

     print("X shape", df_X.shape)
     print("y shape", df_y.shape)
     ✓ 0.0s                                                    Python
```

```
··   X shape (506, 13)
     y shape (506, 1)
```

Questions

- I tried running the last task but am not sure if I am on the right track. Could you please help me review the steps and confirm if everything is correct?

```
### code for step one:
splitter = SklearnDataSplitter(splitter='RepeatedKFold', n_repeats=1, n_splits=10)
splitter.evaluate(X=X,
                  y=df_y,
                  models=[model3],
                  preprocessor=preprocessor,
                  savepath=savepath,
                  verbosity=3)
```

[33] ✓ 35.4s Python

```
Warning: unable to make Scatter.plot_best_worst_per_point plot. Skipping...
Warning: unable to make Scatter.plot_best_worst_per_point plot. Skipping...
```

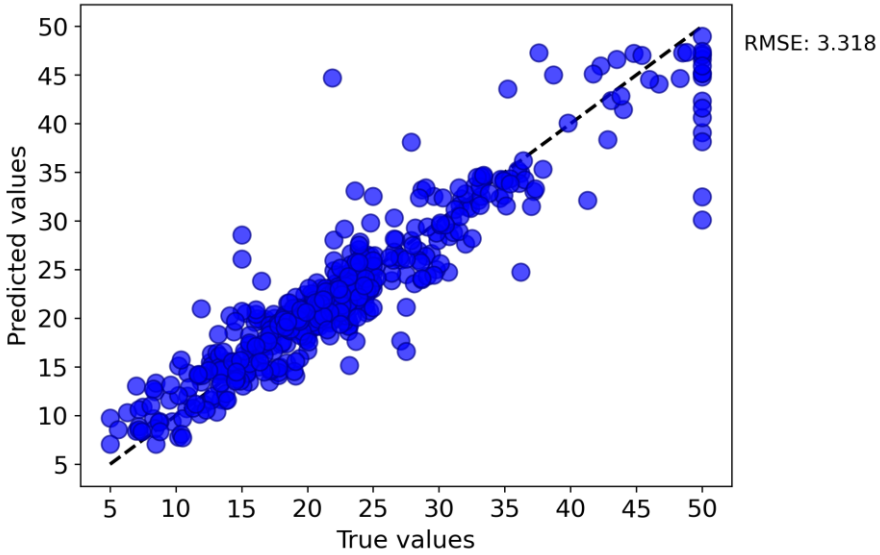
```
### code for step 2:

model4 = SklearnModel(model='RandomForestRegressor', max_depth=3)

# Create the splitter for 10-fold cross-validation with RepeatedKFold
splitter = SklearnDataSplitter(splitter='RepeatedKFold', n_repeats=1, n_splits=10)

# Perform the cross-validation with the new model
splitter.evaluate(X=df_X,
                  y=df_y,
                  models=[model4],
                  preprocessor=preprocessor,
                  savepath=savepath,
                  verbosity=3)
```

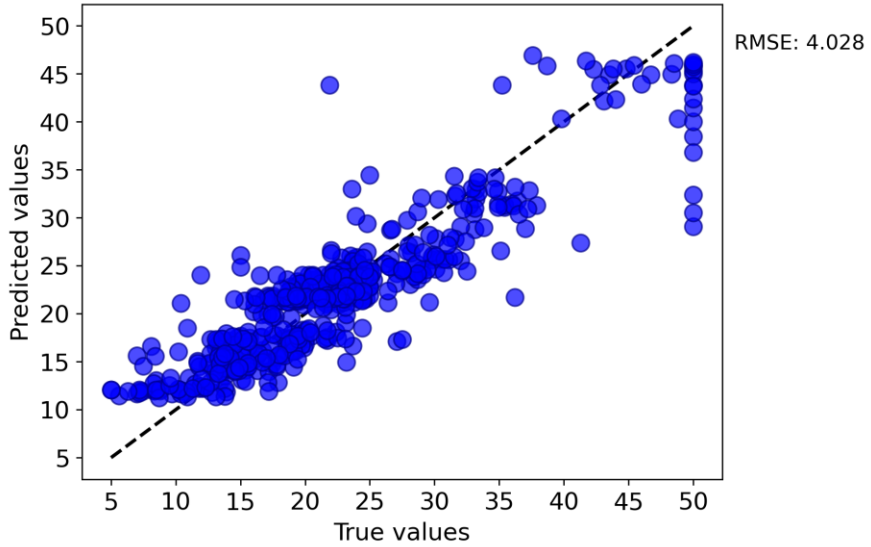
[37] ✓ 31.5s Python

```
Warning: unable to make Scatter.plot_best_worst_per_point plot. Skipping...
Warning: unable to make Scatter.plot_best_worst_per_point plot. Skipping...
```

Questions

- I tried running the last task but am not sure if I am on the right track. Could you please help me review the steps and confirm if everything is correct?





Hours Summary

Date	Hours	Description of Work
07/27/2024	35 min	Complete Introduction to MAST-ML activity