04b_Modeling_Final

April 14, 2023

1 ADS-508-01-SP23 Team 8: Final Project

2 Train model

Much of the code is modified from `Fregly, C., & Barth, A. (2021). Data science on AWS: Implementing end-to-end, continuous AI and machine learning pipelines. O'Reilly.`

2.1 Install missing dependencies

PyAthena is a Python DB API 2.0 (PEP 249) compliant client for Amazon Athena.

[2]:

```
!pip install --disable-pip-version-check -q PyAthena==2.1.0
!pip install --disable-pip-version-check -q sagemaker-experiments==0.1.26
!pip install missingno
!pip install scikit-optimize
```

```
WARNING: The directory '/root/.cache/pip' or its parent directory is not
owned or is not writable by the current user. The cache has been disabled. Check
the permissions and owner of that directory. If executing pip with sudo, you
should use sudo's -H flag.
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager. It is
recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
WARNING: The directory '/root/.cache/pip' or its parent directory is
not owned or is not writable by the current user. The cache has been disabled.
Check the permissions and owner of that directory. If executing pip with sudo,
you should use sudo's -H flag.
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager. It is
recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
WARNING: The directory '/root/.cache/pip' or its parent directory is
not owned or is not writable by the current user. The cache has been disabled.
Check the permissions and owner of that directory. If executing pip with sudo,
you should use sudo's -H flag.
```

```
Collecting missingno
  Downloading missingno-0.5.2-py3-none-any.whl (8.7 kB)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages
(from missingno) (1.21.6)
Requirement already satisfied: seaborn in /opt/conda/lib/python3.7/site-packages
(from missingno) (0.10.0)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.7/site-
packages (from missingno) (3.1.3)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages
(from missingno) (1.4.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->missingno) (1.1.0)
Requirement already satisfied: python-dateutil>=2.1 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/opt/conda/lib/python3.7/site-packages (from matplotlib->missingno) (2.4.6)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-
```

```
packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: pandas>=0.22.0 in /opt/conda/lib/python3.7/site-
packages (from seaborn->missingno) (1.3.5)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages
(from cycler>=0.10->matplotlib->missingno) (1.14.0)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.7/site-
packages (from kiwisolver>=1.0.1->matplotlib->missingno) (59.3.0)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-
packages (from pandas>=0.22.0->seaborn->missingno) (2019.3)
Installing collected packages: missingno
Successfully installed missingno-0.5.2
WARNING: Running pip as the 'root' user can result in broken permissions

and conflicting behaviour with the system package manager. It is recommended to

use a virtual environment instead: https://pip.pypa.io/warnings/venv

WARNING: The directory '/root/.cache/pip' or its parent directory is

not owned or is not writable by the current user. The cache has been disabled.

Check the permissions and owner of that directory. If executing pip with sudo,

you should use sudo's -H flag.

Collecting scikit-optimize
  Downloading scikit_optimize-0.9.0-py2.py3-none-any.whl (100 kB)
                          100.3/100.3 kB
222.4 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn>=0.20.0 in
/opt/conda/lib/python3.7/site-packages (from scikit-optimize) (0.22.1)
Requirement already satisfied: numpy>=1.13.3 in /opt/conda/lib/python3.7/site-
packages (from scikit-optimize) (1.21.6)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-
packages (from scikit-optimize) (1.2.0)
Requirement already satisfied: scipy>=0.19.1 in /opt/conda/lib/python3.7/site-
packages (from scikit-optimize) (1.4.1)
Collecting pyaml>=16.9
  Downloading pyaml-21.10.1-py2.py3-none-any.whl (24 kB)
Requirement already satisfied: PyYAML in /opt/conda/lib/python3.7/site-packages
(from pyaml>=16.9->scikit-optimize) (6.0)
Installing collected packages: pyaml, scikit-optimize
Successfully installed pyaml-21.10.1 scikit-optimize-0.9.0
WARNING: Running pip as the 'root' user can result in broken permissions

and conflicting behaviour with the system package manager. It is recommended to

use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

2.2 Globally import libraries

[3]:

```
import boto3
from botocore.client import ClientError
import pandas as pd
import numpy as np
from pyathena import connect
from IPython.core.display import display, HTML
import missingno as msno
from skopt import BayesSearchCV
from skopt.space import Real, Categorical, Integer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split, cross_val_score,␣
  ↪GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.impute import SimpleImputer
from sklearn.metrics import r2_score, mean_squared_error
from sklearn.linear_model import Lasso
import datetime as dt
import time
import sagemaker
from smexperiments.experiment import Experiment
from smexperiments.trial import Trial
import joblib
import os
from io import BytesIO

%matplotlib inline
```

2.3 Instantiate AWS SageMaker and S3 sessions

[4]:

```
session = boto3.session.Session()
role = sagemaker.get_execution_role()
region = session.region_name
sagemaker_session = sagemaker.Session()
def_bucket = sagemaker_session.default_bucket()
bucket = 'sagemaker-us-east-ads508-sp23-t8'

s3 = boto3.Session().client(service_name="s3",
                            region_name=region)

sm = boto3.Session().client(service_name="sagemaker",
                            region_name=region)
```

```
[5]: print(f"Default bucket: {def_bucket}")
     print(f"Public T8 bucket: {bucket}")
```

```
Default bucket: sagemaker-us-east-1-657724983756
Public T8 bucket: sagemaker-us-east-ads508-sp23-t8
```

2.4 Pass in train and test X from CSV

```
[6]: s3_train_x01_csv_path = f"s3://{def_bucket}/team_8_data/modeling_data/training/
      ↪train_x01.csv"
     train_x01 = pd.read_csv(s3_train_x01_csv_path)
     s3_test_x01_csv_path = f"s3://{def_bucket}/team_8_data/modeling_data/testing/
      ↪test_x01.csv"
     test_x01 = pd.read_csv(s3_test_x01_csv_path)

     print(f'{train_x01.shape}')
     print(f'\n{test_x01.shape}')
```

```
(25284, 48)

(6321, 48)
```

2.5 Pass in train and test y from np array

```
[7]: # Define the S3 object key
     train_y01_s3_key = 'team_8_data/modeling_data/training/train_y01.npy'

     # Load the numpy array from S3
     with BytesIO() as data:
         s3.download_fileobj(def_bucket, train_y01_s3_key, data)
         data.seek(0)
         train_y01 = np.load(data)

     # Define the S3 object key
     test_y01_s3_key = 'team_8_data/modeling_data/testing/test_y01.npy'

     # Load the numpy array from S3
     with BytesIO() as data:
         s3.download_fileobj(def_bucket, test_y01_s3_key, data)
         data.seek(0)
         test_y01 = np.load(data)

     train_y01 = train_y01.ravel()
     test_y01 = test_y01.ravel()

     # Confirm that the numpy array was loaded from S3
     print(f'{train_y01.shape}')
     print(f'{test_y01.shape}')
```

```
(25284,)
(6321,)
```

2.6 Model Training using Grid search with 5-fold cross-validation

2.6.1 Neural Network

[8]:

```
# Start timer script
start_time = dt.datetime.today()

# Citation: Hochberg, 2018; Shanmukh, 2021
m1v1_nn_pip = Pipeline([('si', SimpleImputer(strategy='median')),
                     ('ss', StandardScaler()),
                     ('nn', MLPRegressor(random_state=1699))])

nodes_h = 3
predictors_p = 49

hidden_layer_sizes_hparam = [[100,],
                             [(nodes_h*(predictors_p+1))+nodes_h+1,],
                             [50, 50]
                            ]
activation_hparam = ['logistic', 'relu']
solver_hparam = ['adam']
alpha_hparam = [.0001, .0005, .001]
learn_rate_hparam = ['constant', 'invscaling']

#hidden_layer_sizes_hparam = [[100,]]
#activation_hparam = ['relu']
#solver_hparam = ['adam']
#alpha_hparam = [.0001]
#learn_rate_hparam = ['invscaling']

m1v1_nn_grd = {'nn__hidden_layer_sizes': hidden_layer_sizes_hparam,
            'nn__activation': activation_hparam,
            'nn__solver': solver_hparam,
            'nn__alpha': alpha_hparam,
            'nn__learning_rate': learn_rate_hparam
           }

m1v1_nn = GridSearchCV(m1v1_nn_pip,
                       m1v1_nn_grd,
                       scoring='neg_root_mean_squared_error',
                       n_jobs=2,
                       refit=True,
                       verbose=2)

m1v1_nn.fit(train_x01, train_y01)
```

```
print(f'Best Estimator:\n{m1v1_nn.best_estimator_}')

print(pd.DataFrame(m1v1_nn.cv_results_))

train_m1v1_nn_y01_pred = m1v1_nn.predict(train_x01)
print(train_m1v1_nn_y01_pred)

test_m1v1_nn_y01_pred = m1v1_nn.predict(test_x01)
print(test_m1v1_nn_y01_pred)

# Display evaluation metrics
# R-sq
train_m1v1_nn_r2 = r2_score(train_y01, train_m1v1_nn_y01_pred)
test_m1v1_nn_r2 = r2_score(test_y01, test_m1v1_nn_y01_pred)

print(f'Train R-sq:\n{train_m1v1_nn_r2}')
print(f'Test R-sq:\n{test_m1v1_nn_r2}')

# RMSE
train_m1v1_nn_rmse = mean_squared_error(train_y01, train_m1v1_nn_y01_pred,␣
 ↪squared=False)
test_m1v1_nn_rmse = mean_squared_error(test_y01, test_m1v1_nn_y01_pred,␣
 ↪squared=False)

print(f'Train RMSE:\n{train_m1v1_nn_rmse}')
print(f'Test RMSE:\n{test_m1v1_nn_rmse}')

# End timer script
end_time = dt.datetime.today()
time_elapse = end_time - start_time
print(f'End Time = {end_time}')
print(f'Script Time = {time_elapse}')
```

```
Fitting 5 folds for each of 36 candidates, totalling 180 fits
```

```
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed: 12.0min
[Parallel(n_jobs=2)]: Done 158 tasks      | elapsed: 45.9min
[Parallel(n_jobs=2)]: Done 180 out of 180 | elapsed: 51.7min finished
```

```
Best Estimator:
Pipeline(memory=None,
         steps=[('si',
                 SimpleImputer(add_indicator=False, copy=True, fill_value=None,
                               missing_values=nan, strategy='median',
                               verbose=0)),
                ('ss',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('nn',
                 MLPRegressor(activation='relu', alpha=0.0005,
                              batch_size='auto', beta_1=0.9, beta_2=0.999,
                              early_stopping=False, epsilon=1e-08,
                              hidden_layer_sizes=[50, 50],
                              learning_rate='constant',
                              learning_rate_init=0.001, max_fun=15000,
                              max_iter=200, momentum=0.9, n_iter_no_change=10,
                              nesterovs_momentum=True, power_t=0.5,
                              random_state=1699, shuffle=True, solver='adam',
                              tol=0.0001, validation_fraction=0.1,
                              verbose=False, warm_start=False))],
         verbose=False)
    mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       33.143548      0.529329         0.032452        0.003388
1       34.235025      0.891932         0.034432        0.003698
2       45.957392      0.721042         0.042339        0.000732
3       46.024776      0.781211         0.050556        0.017036
4       38.072548      0.445525         0.033994        0.003400
5       36.995700      0.322889         0.032575        0.000501
6       32.707927      0.229827         0.032212        0.000665
7       32.493836      0.077391         0.032501        0.000327
8       45.085183      0.421935         0.044698        0.003383
9       45.192096      0.353731         0.045635        0.003949
10      37.167754      0.503359         0.034419        0.004507
11      37.187461      0.628572         0.033794        0.003179
12      32.951811      0.651863         0.032234        0.000757
13      32.465124      0.270572         0.032322        0.000358
14      47.043259      2.372003         0.042367        0.000092
15      45.504390      0.374914         0.042963        0.001024
16      37.119073      0.681981         0.033486        0.001257
17      37.297852      0.599174         0.032584        0.000332
18      25.550653      0.229620         0.020148        0.003253
19      25.787540      0.218238         0.017209        0.000411
20      33.771521      0.603930         0.024322        0.009828
21      33.597681      0.430749         0.019367        0.000481
22      31.016342      0.396437         0.017053        0.001032
23      31.029871      0.554800         0.018221        0.001761
24      26.011305      0.443071         0.017106        0.000348
25      25.785980      0.471428         0.016555        0.001151
26      33.899021      0.349528         0.020482        0.001773
27      33.563256      0.528809         0.019615        0.000963
28      31.141742      0.566396         0.020337        0.006852
29      30.997358      0.365988         0.017249        0.000827
30      25.684840      0.256868         0.017028        0.000479
31      25.798281      0.302832         0.017244        0.000354
32      33.996955      0.638432         0.019921        0.000500
```

```
33      33.543362      0.277484         0.019681        0.000308
34      30.799481      0.347098         0.017663        0.000995
35      30.831124      0.491073         0.015790        0.002267

   param_nn__activation param_nn__alpha param_nn__hidden_layer_sizes  \
0              logistic          0.0001                        [100]
1              logistic          0.0001                        [100]
2              logistic          0.0001                        [154]
3              logistic          0.0001                        [154]
4              logistic          0.0001                     [50, 50]
5              logistic          0.0001                     [50, 50]
6              logistic          0.0005                        [100]
7              logistic          0.0005                        [100]
8              logistic          0.0005                        [154]
9              logistic          0.0005                        [154]
10             logistic          0.0005                     [50, 50]
11             logistic          0.0005                     [50, 50]
12             logistic           0.001                        [100]
13             logistic           0.001                        [100]
14             logistic           0.001                        [154]
15             logistic           0.001                        [154]
16             logistic           0.001                     [50, 50]
17             logistic           0.001                     [50, 50]
18                 relu          0.0001                        [100]
19                 relu          0.0001                        [100]
20                 relu          0.0001                        [154]
21                 relu          0.0001                        [154]
22                 relu          0.0001                     [50, 50]
23                 relu          0.0001                     [50, 50]
24                 relu          0.0005                        [100]
25                 relu          0.0005                        [100]
26                 relu          0.0005                        [154]
27                 relu          0.0005                        [154]
28                 relu          0.0005                     [50, 50]
29                 relu          0.0005                     [50, 50]
30                 relu           0.001                        [100]
31                 relu           0.001                        [100]
32                 relu           0.001                        [154]
33                 relu           0.001                        [154]
34                 relu           0.001                     [50, 50]
35                 relu           0.001                     [50, 50]

   param_nn__learning_rate param_nn__solver  \
0                 constant             adam
1               invscaling             adam
2                 constant             adam
3               invscaling             adam
4                 constant             adam
```

```
5                 invscaling             adam
6                   constant             adam
7                 invscaling             adam
8                   constant             adam
9                 invscaling             adam
10                  constant             adam
11                invscaling             adam
12                  constant             adam
13                invscaling             adam
14                  constant             adam
15                invscaling             adam
16                  constant             adam
17                invscaling             adam
18                  constant             adam
19                invscaling             adam
20                  constant             adam
21                invscaling             adam
22                  constant             adam
23                invscaling             adam
24                  constant             adam
25                invscaling             adam
26                  constant             adam
27                invscaling             adam
28                  constant             adam
29                invscaling             adam
30                  constant             adam
31                invscaling             adam
32                  constant             adam
33                invscaling             adam
34                  constant             adam
35                invscaling             adam

                                               params  split0_test_score  \
0   {'nn__activation': 'logistic', 'nn__alpha': 0…          -7.886156
1   {'nn__activation': 'logistic', 'nn__alpha': 0…          -7.886156
2   {'nn__activation': 'logistic', 'nn__alpha': 0…          -6.666789
3   {'nn__activation': 'logistic', 'nn__alpha': 0…          -6.666789
4   {'nn__activation': 'logistic', 'nn__alpha': 0…          -4.942271
5   {'nn__activation': 'logistic', 'nn__alpha': 0…          -4.942271
6   {'nn__activation': 'logistic', 'nn__alpha': 0…          -7.885017
7   {'nn__activation': 'logistic', 'nn__alpha': 0…          -7.885017
8   {'nn__activation': 'logistic', 'nn__alpha': 0…          -6.660854
9   {'nn__activation': 'logistic', 'nn__alpha': 0…          -6.660854
10  {'nn__activation': 'logistic', 'nn__alpha': 0…          -4.943635
11  {'nn__activation': 'logistic', 'nn__alpha': 0…          -4.943635
12  {'nn__activation': 'logistic', 'nn__alpha': 0…          -7.886704
13  {'nn__activation': 'logistic', 'nn__alpha': 0…          -7.886704
14  {'nn__activation': 'logistic', 'nn__alpha': 0…          -6.656225
```

```
15  {'nn__activation': 'logistic', 'nn__alpha': 0…          -6.656225
16  {'nn__activation': 'logistic', 'nn__alpha': 0…          -4.945009
17  {'nn__activation': 'logistic', 'nn__alpha': 0…          -4.945009
18  {'nn__activation': 'relu', 'nn__alpha': 0.0001…          -4.755215
19  {'nn__activation': 'relu', 'nn__alpha': 0.0001…          -4.755215
20  {'nn__activation': 'relu', 'nn__alpha': 0.0001…          -3.408820
21  {'nn__activation': 'relu', 'nn__alpha': 0.0001…          -3.408820
22  {'nn__activation': 'relu', 'nn__alpha': 0.0001…          -2.138443
23  {'nn__activation': 'relu', 'nn__alpha': 0.0001…          -2.138443
24  {'nn__activation': 'relu', 'nn__alpha': 0.0005…          -4.793217
25  {'nn__activation': 'relu', 'nn__alpha': 0.0005…          -4.793217
26  {'nn__activation': 'relu', 'nn__alpha': 0.0005…          -3.415002
27  {'nn__activation': 'relu', 'nn__alpha': 0.0005…          -3.415002
28  {'nn__activation': 'relu', 'nn__alpha': 0.0005…          -2.072638
29  {'nn__activation': 'relu', 'nn__alpha': 0.0005…          -2.072638
30  {'nn__activation': 'relu', 'nn__alpha': 0.001,…          -4.732671
31  {'nn__activation': 'relu', 'nn__alpha': 0.001,…          -4.732671
32  {'nn__activation': 'relu', 'nn__alpha': 0.001,…          -3.442814
33  {'nn__activation': 'relu', 'nn__alpha': 0.001,…          -3.442814
34  {'nn__activation': 'relu', 'nn__alpha': 0.001,…          -2.158172
35  {'nn__activation': 'relu', 'nn__alpha': 0.001,…          -2.158172

    split1_test_score  split2_test_score  split3_test_score  \
0           -8.056679          -7.571552          -7.917553
1           -8.056679          -7.571552          -7.917553
2           -6.611240          -6.292724          -6.284235
3           -6.611240          -6.292724          -6.284235
4           -4.733368          -4.564217          -4.642696
5           -4.733368          -4.564217          -4.642696
6           -8.054838          -7.574916          -7.917823
7           -8.054838          -7.574916          -7.917823
8           -6.610766          -6.290245          -6.284635
9           -6.610766          -6.290245          -6.284635
10          -4.732347          -4.564699          -4.641053
11          -4.732347          -4.564699          -4.641053
12          -8.054588          -7.577553          -7.916608
13          -8.054588          -7.577553          -7.916608
14          -6.610904          -6.288636          -6.287919
15          -6.610904          -6.288636          -6.287919
16          -4.731240          -4.566586          -4.652802
17          -4.731240          -4.566586          -4.652802
18          -4.727183          -4.493818          -4.706454
19          -4.727183          -4.493818          -4.706454
20          -3.261693          -3.143096          -3.177231
21          -3.261693          -3.143096          -3.177231
22          -2.074308          -1.988741          -2.209510
23          -2.074308          -1.988741          -2.209510
24          -4.620354          -4.531985          -4.668786
```

```
25          -4.620354          -4.531985          -4.668786
26          -3.165363          -3.111582          -3.153980
27          -3.165363          -3.111582          -3.153980
28          -1.978748          -1.917381          -2.099757
29          -1.978748          -1.917381          -2.099757
30          -4.652343          -4.511500          -4.645985
31          -4.652343          -4.511500          -4.645985
32          -3.205790          -3.099954          -3.160798
33          -3.205790          -3.099954          -3.160798
34          -1.995288          -1.845477          -2.203662
35          -1.995288          -1.845477          -2.203662

    split4_test_score  mean_test_score  std_test_score  rank_test_score
0           -7.868349        -7.860058        0.158714               31
1           -7.868349        -7.860058        0.158714               31
2           -6.383989        -6.447796        0.160961               29
3           -6.383989        -6.447796        0.160961               29
4           -4.418050        -4.660120        0.174970               19
5           -4.418050        -4.660120        0.174970               19
6           -7.869608        -7.860440        0.157032               33
7           -7.869608        -7.860440        0.157032               33
8           -6.380977        -6.445496        0.159897               27
9           -6.380977        -6.445496        0.159897               27
10          -4.419229        -4.660193        0.174983               21
11          -4.419229        -4.660193        0.174983               21
12          -7.871709        -7.861432        0.156003               35
13          -7.871709        -7.861432        0.156003               35
14          -6.378792        -6.444495        0.158522               25
15          -6.378792        -6.444495        0.158522               25
16          -4.421981        -4.663523        0.174169               23
17          -4.421981        -4.663523        0.174169               23
18          -4.317670        -4.600068        0.168870               17
19          -4.317670        -4.600068        0.168870               17
20          -3.227816        -3.243731        0.092065               11
21          -3.227816        -3.243731        0.092065               11
22          -1.943277        -2.070856        0.096772                5
23          -1.943277        -2.070856        0.096772                5
24          -4.332465        -4.589361        0.153697               15
25          -4.332465        -4.589361        0.153697               15
26          -3.171301        -3.203446        0.107820                7
27          -3.171301        -3.203446        0.107820                7
28          -2.024192        -2.018543        0.065378                1
29          -2.024192        -2.018543        0.065378                1
30          -4.367697        -4.582040        0.128561               13
31          -4.367697        -4.582040        0.128561               13
32          -3.234924        -3.228856        0.116271                9
33          -3.234924        -3.228856        0.116271                9
34          -2.029527        -2.046425        0.126879                3
```

```
35          -2.029527        -2.046425        0.126879                 3
[ 5.77257103  2.71321492 34.23584791 …  2.24031845 14.15423095
  0.35154327]
[41.00555646 -0.83261161 37.19630089 … -1.23440346  6.85440592
 17.10392427]
Train R-sq:
0.9953169265206908
Test R-sq:
0.9946084935331966
Train RMSE:
1.2934970735367237
Test RMSE:
1.4032555490004501
End Time = 2023-04-13 19:29:37.552207
Script Time = 0:52:22.942444
```

```
/opt/conda/lib/python3.7/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:571:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

[9]:

```python
s3_m1v1_nn_pqt_base_path = f"../models"

if not os.path.exists(s3_m1v1_nn_pqt_base_path):
    os.makedirs(s3_m1v1_nn_pqt_base_path)

s3_m1v1_nn_pqt_path = os.path.join(s3_m1v1_nn_pqt_base_path,
                                   'm1v1_nn.parquet')

# save the model to disk using joblib
joblib.dump(m1v1_nn,
            s3_m1v1_nn_pqt_path)

# load the saved model from disk using joblib
m1v1_nn_fitted = joblib.load(s3_m1v1_nn_pqt_path)
```

[10]:

```python
# specify the S3 bucket and key where you want to save the model
m1v1_nn_key_name = 'team_8_data/models/m1v1_nn.parquet'

# save the model to an in-memory buffer
buffer = BytesIO()
joblib.dump(m1v1_nn, buffer)

# upload the buffer to S3
buffer.seek(0)
s3.upload_fileobj(buffer, def_bucket, m1v1_nn_key_name)
```

```
# load the saved model from S3
#buffer = BytesIO()
#s3.download_fileobj(def_bucket, m1v1_nn_key_name, buffer)
#buffer.seek(0)
#m1v1_nn_fitted = joblib.load(buffer)
```

2.6.2 Lasso - Using GridSearchCV

[11]:

```
# Start timer script
start_time = dt.datetime.today()

# Citation: Hochberg, 2018; Shanmukh, 2021
m2v1_ls_pip = Pipeline([('si', SimpleImputer(strategy='median')),
                        ('ss', StandardScaler()),
                        ('ls', Lasso(random_state=1699))])

alpha_hparam = [.01, .05, .1, .5, 1, 2]
selection_hparam = ['cyclic', 'random']


m2v1_ls_grd = {'ls__alpha': alpha_hparam,
               'ls__selection': selection_hparam
           }

m2v1_ls = GridSearchCV(m2v1_ls_pip,
                       m2v1_ls_grd,
                       scoring='neg_root_mean_squared_error',
                       n_jobs=2,
                       refit=True,
                       verbose=2)

m2v1_ls.fit(train_x01, train_y01)

print(f'Best Estimator:\n{m2v1_ls.best_estimator_}')
print(f'Coefficients:\n{m2v1_ls.best_estimator_.named_steps["ls"].coef_}')

print(pd.DataFrame(m2v1_ls.cv_results_))

train_m2v1_ls_y01_pred = m2v1_ls.predict(train_x01)
print(train_m2v1_ls_y01_pred)

test_m2v1_ls_y01_pred = m2v1_ls.predict(test_x01)
print(test_m2v1_ls_y01_pred)

# Display evaluation metrics
# R-sq
train_m2v1_ls_r2 = r2_score(train_y01, train_m2v1_ls_y01_pred)
```

```python
test_m2v1_ls_r2 = r2_score(test_y01, test_m2v1_ls_y01_pred)

print(f'Train R-sq:\n{train_m2v1_ls_r2}')
print(f'Test R-sq:\n{test_m2v1_ls_r2}')

# RMSE
train_m2v1_ls_rmse = mean_squared_error(train_y01, train_m2v1_ls_y01_pred,␣
 ↪squared=False)
test_m2v1_ls_rmse = mean_squared_error(test_y01, test_m2v1_ls_y01_pred,␣
 ↪squared=False)

print(f'Train RMSE:\n{train_m2v1_ls_rmse}')
print(f'Test RMSE:\n{test_m2v1_ls_rmse}')

# End timer script
end_time = dt.datetime.today()
time_elapse = end_time - start_time
print(f'End Time = {end_time}')
print(f'Script Time = {time_elapse}')
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done  37 tasks      | elapsed:   15.4s
[Parallel(n_jobs=2)]: Done  60 out of  60 | elapsed:   18.6s finished

Best Estimator:
Pipeline(memory=None,
         steps=[('si',
                 SimpleImputer(add_indicator=False, copy=True, fill_value=None,
                               missing_values=nan, strategy='median',
                               verbose=0)),
                ('ss',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('ls',
                 Lasso(alpha=0.01, copy_X=True, fit_intercept=True,
                       max_iter=1000, normalize=False, positive=False,
                       precompute=False, random_state=1699, selection='random',
                       tol=0.0001, warm_start=False))],
         verbose=False)
Coefficients:
[-0.04698528  1.1655262   0.         -1.02395897  1.13308637 -0.
 -0.          0.          0.02814286 -0.         -0.0307876   0.
  0.         -0.         -0.01445511  0.         -0.          2.92859099
  2.50110937  5.60822761  1.40168756 -2.34296164 -1.18943262 -0.33822063
 -1.3657179  -3.4737577  -6.79353668  0.12912462  0.20841108 -0.04818647
 -0.51860673  2.15041078  0.72946348 -0.44590033  0.         -1.47779489
 -0.48312123  0.          1.54018429  0.49113008  0.56200594 -0.72542784
 -8.16308228  0.40673301 -1.2849073   0.46802939  0.73592459  1.6206294 ]
    mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0        1.025094      0.110517         0.009961        0.001149   
1        1.579347      0.095895         0.010434        0.000936   
2        0.512077      0.018247         0.010032        0.001147   
3        0.840451      0.166026         0.009896        0.001420   
4        0.431524      0.015982         0.008961        0.001188   
5        1.213542      0.502992         0.010119        0.001508   
6        0.287477      0.024440         0.008343        0.000368   
7        0.303767      0.005579         0.010572        0.001484   
8        0.253030      0.003281         0.010716        0.000799   
9        0.258238      0.008042         0.010476        0.000802   
10       0.242289      0.005520         0.010471        0.000613   
11       0.238470      0.028518         0.008936        0.001677   

   param_ls__alpha param_ls__selection  \
0             0.01              cyclic   
1             0.01              random   
2             0.05              cyclic   
3             0.05              random   
4              0.1              cyclic   
5              0.1              random   
6              0.5              cyclic   
7              0.5              random   
8                1              cyclic   
9                1              random   
10               2              cyclic   
11               2              random   

                                            params  split0_test_score  \
0   {'ls__alpha': 0.01, 'ls__selection': 'cyclic'}         -11.378789   
1   {'ls__alpha': 0.01, 'ls__selection': 'random'}         -11.378654   
2   {'ls__alpha': 0.05, 'ls__selection': 'cyclic'}         -11.381059   
3   {'ls__alpha': 0.05, 'ls__selection': 'random'}         -11.380958   
4    {'ls__alpha': 0.1, 'ls__selection': 'cyclic'}         -11.396508   
5    {'ls__alpha': 0.1, 'ls__selection': 'random'}         -11.396574   
6    {'ls__alpha': 0.5, 'ls__selection': 'cyclic'}         -11.806394   
7    {'ls__alpha': 0.5, 'ls__selection': 'random'}         -11.806288   
8      {'ls__alpha': 1, 'ls__selection': 'cyclic'}         -12.008540   
9      {'ls__alpha': 1, 'ls__selection': 'random'}         -12.008700   
10     {'ls__alpha': 2, 'ls__selection': 'cyclic'}         -12.496541   
11     {'ls__alpha': 2, 'ls__selection': 'random'}         -12.496534   

    split1_test_score  split2_test_score  split3_test_score  \
0          -11.391382         -11.411164         -11.505775   
1          -11.391378         -11.411135         -11.505295   
2          -11.384965         -11.423042         -11.519926   
3          -11.384860         -11.423228         -11.520242   
```

```
4          -11.408320         -11.450277         -11.549378
5          -11.408500         -11.450586         -11.550226
6          -11.866156         -11.935607         -12.014084
7          -11.866341         -11.935859         -12.014298
8          -12.112724         -12.186260         -12.226056
9          -12.112880         -12.186494         -12.226203
10         -12.607476         -12.682111         -12.710059
11         -12.607472         -12.682084         -12.710063

    split4_test_score  mean_test_score  std_test_score  rank_test_score
0          -11.386550       -11.414732        0.046761                2
1          -11.386614       -11.414615        0.046588                1
2          -11.395211       -11.420840        0.051671                3
3          -11.395450       -11.420948        0.051800                4
4          -11.417847       -11.444466        0.055417                5
5          -11.418167       -11.444811        0.055679                6
6          -11.850882       -11.894625        0.072744                7
7          -11.851056       -11.894768        0.072833                8
8          -12.081994       -12.123115        0.076825                9
9          -12.082092       -12.123274        0.076840               10
10         -12.596496       -12.618537        0.074712               11
11         -12.596565       -12.618544        0.074706               12
[ 1.00885432  2.1155105  20.65181841 …  5.04395528 20.70033313
  9.2996565 ]
[43.29645545  3.29328038 36.54323059 … -0.46893988  1.17345827
  3.2687973 ]
Train R-sq:
0.6369229292317483
Test R-sq:
0.6479556475979478
Train RMSE:
11.389361452679568
Test RMSE:
11.339147217962033
End Time = 2023-04-13 19:29:57.540560
Script Time = 0:00:19.711830
```

[12]:
```
coef_intercept = np.hstack((m2v1_ls.best_estimator_.named_steps["ls"].coef_,
                            m2v1_ls.best_estimator_.named_steps["ls"].
 ↪intercept_))
#print(coef_intercept)

coef_intercept_df01 = pd.DataFrame(coef_intercept)
#display(coef_intercept_df01)

train_x01_col_names = list(train_x01.columns)
train_x01_col_names.append('intercept')
```

```
train_x01_col_names_df01 = pd.DataFrame(train_x01_col_names)
#display(train_x01_col_names_df01)

model_params = pd.concat([train_x01_col_names_df01, coef_intercept_df01],␣
 ↪axis=1)
display(model_params)
```

```
                             0          0
0                borough_bronx  -0.046985
1             borough_brooklyn   1.165526
2            borough_manhattan   0.000000
3               borough_queens  -1.023959
4        borough_staten island   1.133086
5        relative_data_year_-4  -0.000000
6        relative_data_year_-3  -0.000000
7        relative_data_year_-2   0.000000
8        relative_data_year_-1   0.028143
9         relative_data_year_0  -0.000000
10       complaint_type_FELONY  -0.030788
11  complaint_type_MISDEMEANOR   0.000000
12    complaint_type_VIOLATION   0.000000
13  annual_evictions_x_borough  -0.000000
14     annual_complaint_counts  -0.014455
15               annual_grad_n   0.000000
16        annual_dropped_out_n  -0.000000
17                    totalpop   2.928591
18                         men   2.501109
19                       women   5.608228
20                    hispanic   1.401688
21                       white  -2.342962
22                       black  -1.189433
23                      native  -0.338221
24                       asian  -1.365718
25                     citizen  -3.473758
26                      income  -6.793537
27                   incomeerr   0.129125
28                incomepercap   0.208411
29             incomepercaperr  -0.048186
30                professional  -0.518607
31                     service   2.150411
32                      office   0.729463
33                construction  -0.445900
34                  production   0.000000
35                       drive  -1.477795
36                     carpool  -0.483121
37                     transit   0.000000
38                        walk   1.540184
```

```
39                 othertransp   0.491130
40                  workathome   0.562006
41                 meancommute  -0.725428
42                    employed  -8.163082
43                 privatework   0.406733
44                  publicwork  -1.284907
45                selfemployed   0.468029
46                  familywork   0.735925
47                unemployment   1.620629
48                   intercept  24.475273
```

[13]:

```
# specify the S3 bucket and key where you want to save the model
m2v1_ls_key_name = 'team_8_data/models/m2v1_ls.parquet'

# save the model to an in-memory buffer
buffer = BytesIO()
joblib.dump(m2v1_ls, buffer)

# upload the buffer to S3
buffer.seek(0)
s3.upload_fileobj(buffer, def_bucket, m2v1_ls_key_name)

# load the saved model from S3
#buffer = BytesIO()
#s3.download_fileobj(def_bucket, m2v1_ls_key_name, buffer)
#buffer.seek(0)
#m2v1_ls_fitted = joblib.load(buffer)
```

2.6.3 Lasso - Using BayesSearchCV

[14]:

```
# Start timer script
start_time = dt.datetime.today()

# Citation: Hochberg, 2018; Shanmukh, 2021
m2v2_ls_pip = Pipeline([('si', SimpleImputer(strategy='median')),
                        ('ss', StandardScaler()),
                        ('ls', Lasso(random_state=1699))])

alpha_hparam = Real(1e-3, 1e3, prior='log-uniform')
selection_hparam = Categorical(['cyclic', 'random'])
max_iter_hparam = Integer(100, 5000, prior='log-uniform')
warm_start_hparam = Categorical([False, True])


m2v2_ls_grd = {'ls__alpha': alpha_hparam,
               'ls__selection': selection_hparam,
               'ls__max_iter': max_iter_hparam,
               'ls__warm_start': warm_start_hparam
           }

m2v2_ls = BayesSearchCV(m2v2_ls_pip,
                       m2v2_ls_grd,
                       scoring='neg_root_mean_squared_error',
                        cv=5,
                       n_jobs=2,
                       refit=True,
                       verbose=2)

m2v2_ls.fit(train_x01, train_y01)

print(f'Best Estimator:\n{m2v2_ls.best_estimator_}')
print(f'Coefficients:\n{m2v2_ls.best_estimator_.named_steps["ls"].coef_}')

print(pd.DataFrame(m2v2_ls.cv_results_))

train_m2v2_ls_y01_pred = m2v2_ls.predict(train_x01)
print(train_m2v2_ls_y01_pred)

test_m2v2_ls_y01_pred = m2v2_ls.predict(test_x01)
print(test_m2v2_ls_y01_pred)

# Display evaluation metrics
# R-sq
train_m2v2_ls_r2 = r2_score(train_y01, train_m2v2_ls_y01_pred)
test_m2v2_ls_r2 = r2_score(test_y01, test_m2v2_ls_y01_pred)

print(f'Train R-sq:\n{train_m2v2_ls_r2}')
print(f'Test R-sq:\n{test_m2v2_ls_r2}')

# RMSE
train_m2v2_ls_rmse = mean_squared_error(train_y01, train_m2v2_ls_y01_pred,␣
  ↪squared=False)
test_m2v2_ls_rmse = mean_squared_error(test_y01, test_m2v2_ls_y01_pred,␣
  ↪squared=False)

print(f'Train RMSE:\n{train_m2v2_ls_rmse}')
print(f'Test RMSE:\n{test_m2v2_ls_rmse}')

# End timer script
end_time = dt.datetime.today()
time_elapse = end_time - start_time
print(f'End Time = {end_time}')
print(f'Script Time = {time_elapse}')
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s finished
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.6s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.6s finished
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s finished
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.8s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.8s finished
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.3s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.3s finished
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    3.4s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    3.4s finished
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.4s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.4s finished
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s finished
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits
```

```
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    0.7s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.6s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.6s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.7s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.7s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.3s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.3s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   14.7s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   14.7s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.9s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.9s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    7.5s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    7.5s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.3s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.3s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    4.7s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    4.7s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    8.4s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    8.4s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.3s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.3s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    6.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    6.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   26.6s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   26.6s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits
```

```
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   10.9s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   10.9s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    5.7s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    5.7s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   16.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   16.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    4.2s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    4.2s finished
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    3.5s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    3.5s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    8.4s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    8.4s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.4s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.4s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.9s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.9s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    3.8s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    3.8s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits
```

```
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   14.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:   14.1s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.0s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    2.0s finished

Fitting 5 folds for each of 1 candidates, totalling 5 fits

[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=2)]: Done   5 out of   5 | elapsed:    1.1s finished

Best Estimator:
Pipeline(memory=None,
         steps=[('si',
                 SimpleImputer(add_indicator=False, copy=True, fill_value=None,
                               missing_values=nan, strategy='median',
                               verbose=0)),
                ('ss',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('ls',
                 Lasso(alpha=0.00312004499292689, copy_X=True,
                       fit_intercept=True, max_iter=5000, normalize=False,
                       positive=False, precompute=False, random_state=1699,
                       selection='cyclic', tol=0.0001, warm_start=False))],
         verbose=False)
Coefficients:
[-5.90772722e-02  1.20356958e+00  0.00000000e+00 -1.03729787e+00
  1.13319125e+00 -0.00000000e+00 -0.00000000e+00  2.94787994e-03
  3.38749310e-02 -2.05060234e-04 -2.82296287e-02  2.06826899e-02
  0.00000000e+00 -0.00000000e+00 -4.34594017e-02  0.00000000e+00
 -0.00000000e+00  8.48098006e+00  0.00000000e+00  2.76832909e+00
  6.97559873e-01 -3.30318362e+00 -2.14827171e+00 -3.71075424e-01
 -1.90796058e+00 -3.65510811e+00 -6.88025498e+00  1.54667377e-01
  4.43630919e-01 -1.76161247e-01 -1.81767382e+00  1.33115461e+00
  2.87452875e-01 -7.98665794e-01 -3.86055676e-01 -1.45597342e+00
 -4.86681209e-01  0.00000000e+00  1.52719040e+00  4.85589992e-01
  5.68145407e-01 -7.49278734e-01 -8.29828524e+00  0.00000000e+00
 -1.69877386e+00  2.42544215e-01  7.16688567e-01  1.60868297e+00]
   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       0.252393      0.044313         0.008819        0.001531
1       0.211456      0.029397         0.009242        0.001724
2       0.238254      0.037010         0.010291        0.002250
3       0.241629      0.017929         0.010836        0.002084
4       0.371777      0.079431         0.015980        0.004156
5       1.241521      0.785646         0.011383        0.002698
```

```
6        0.476834      0.089144         0.009877        0.001961
7        0.227144      0.035521         0.009860        0.002405
8        0.213959      0.030165         0.009917        0.002330
9        0.243137      0.040122         0.009101        0.001723
10       0.543308      0.085319         0.009564        0.001922
11       0.606364      0.112065         0.008954        0.001687
12       0.449924      0.079588         0.010595        0.003273
13       5.474890      2.152558         0.010317        0.002130
14       1.031375      0.176444         0.009314        0.001718
15       2.952774      1.177147         0.010025        0.002432
16       0.334927      0.052971         0.009441        0.001776
17       0.371395      0.064502         0.009407        0.001627
18       0.370219      0.069268         0.008443        0.001334
19       0.356148      0.062020         0.009524        0.002007
20       0.456968      0.099663         0.013050        0.004496
21       1.705872      0.324273         0.010758        0.004344
22       0.355035      0.064276         0.008949        0.001570
23       3.311975      0.950526         0.010320        0.002188
24       0.873525      0.198976         0.010325        0.003037
25       2.112508      0.328161         0.008992        0.002086
26       9.484036      1.248022         0.010280        0.001913
27       0.354692      0.058464         0.009997        0.001848
28       0.367649      0.061606         0.009237        0.001774
29       0.378164      0.070533         0.009635        0.002129
30       4.151578      2.864317         0.010427        0.002207
31       0.683975      0.111741         0.008902        0.001899
32       0.352353      0.063039         0.009234        0.001608
33       0.376071      0.064972         0.009555        0.001731
34       0.367897      0.065547         0.009819        0.002583
35       2.044964      0.413021         0.009135        0.002086
36       6.089054      1.929319         0.010797        0.002346
37       1.577792      0.276699         0.010048        0.001905
38       1.364861      0.264036         0.010090        0.001880
39       3.132879      0.511682         0.009999        0.002032
40       0.367394      0.062305         0.008894        0.001489
41       0.473200      0.077456         0.010350        0.002553
42       0.362873      0.056866         0.009705        0.002118
43       1.114455      0.251687         0.010146        0.002999
44       0.342149      0.052996         0.009398        0.001434
45       0.369063      0.061133         0.009580        0.001755
46       1.316128      0.174785         0.009341        0.001797
47       4.643917      1.957186         0.010486        0.002173
48       0.740760      0.162634         0.009555        0.002145
49       0.371850      0.062724         0.009669        0.001504

   param_ls__alpha param_ls__max_iter param_ls__selection  \
0         0.919229                372              cyclic
1        20.785016                193              cyclic
```

2	1.469648	441	random
3	9.380698	174	random
4	520.400979	2697	cyclic
5	0.002528	165	random
6	0.252225	949	cyclic
7	2.444393	4088	cyclic
8	107.97026	357	cyclic
9	0.859011	467	cyclic
10	0.020092	5000	cyclic
11	0.003409	246	random
12	0.056266	5000	cyclic
13	0.001	5000	cyclic
14	0.008623	5000	cyclic
15	0.001422	5000	cyclic
16	0.10612	100	random
17	0.001179	100	random
18	0.005593	100	cyclic
19	0.030282	100	random
20	0.012457	100	random
21	0.006117	5000	random
22	0.040493	100	cyclic
23	0.00312	5000	cyclic
24	0.040469	5000	random
25	0.004728	5000	cyclic
26	0.001	5000	random
27	0.061997	100	random
28	0.016873	100	random
29	0.001819	100	random
30	0.00183	5000	cyclic
31	0.013661	5000	cyclic
32	0.046368	100	random
33	0.006918	100	random
34	0.001	100	cyclic
35	0.004217	5000	random
36	0.001	5000	cyclic
37	0.010632	5000	random
38	0.015439	5000	random
39	0.002274	5000	random
40	0.002292	100	cyclic
41	0.033243	5000	cyclic
42	0.009614	100	cyclic
43	0.023583	5000	random
44	0.080033	100	cyclic
45	0.026002	100	cyclic
46	0.007288	5000	cyclic
47	0.001158	5000	cyclic
48	0.080142	5000	random
49	0.003943	100	cyclic

```
   param_ls__warm_start                                          params  \
0                  True  {'ls__alpha': 0.9192293938041608, 'ls__max_ite…
1                  True  {'ls__alpha': 20.785016353070272, 'ls__max_ite…
2                  True  {'ls__alpha': 1.4696478744138841, 'ls__max_ite…
3                  True  {'ls__alpha': 9.380698318075792, 'ls__max_iter…
4                  True  {'ls__alpha': 520.4009792978661, 'ls__max_iter…
5                  True  {'ls__alpha': 0.0025276741815852596, 'ls__max_…
6                  True  {'ls__alpha': 0.2522250210775078, 'ls__max_ite…
7                  True  {'ls__alpha': 2.4443925994929305, 'ls__max_ite…
8                 False  {'ls__alpha': 107.97025991821005, 'ls__max_ite…
9                 False  {'ls__alpha': 0.8590108805849646, 'ls__max_ite…
10                False  {'ls__alpha': 0.020091744449482356, 'ls__max_i…
11                 True  {'ls__alpha': 0.003409256952063032, 'ls__max_i…
12                 True  {'ls__alpha': 0.056266004320859396, 'ls__max_i…
13                False  {'ls__alpha': 0.001, 'ls__max_iter': 5000, 'ls…
14                 True  {'ls__alpha': 0.008623045081308458, 'ls__max_i…
15                False  {'ls__alpha': 0.0014221967527329972, 'ls__max_…
16                False  {'ls__alpha': 0.10611954673637193, 'ls__max_it…
17                 True  {'ls__alpha': 0.0011791282733941873, 'ls__max_…
18                False  {'ls__alpha': 0.005592992209800713, 'ls__max_i…
19                 True  {'ls__alpha': 0.030281735565660575, 'ls__max_i…
20                False  {'ls__alpha': 0.012457400581715239, 'ls__max_i…
21                False  {'ls__alpha': 0.006117246602685628, 'ls__max_i…
22                False  {'ls__alpha': 0.04049343717819463, 'ls__max_it…
23                False  {'ls__alpha': 0.00312004499292689, 'ls__max_it…
24                False  {'ls__alpha': 0.04046875004317316, 'ls__max_it…
25                 True  {'ls__alpha': 0.004728162592110137, 'ls__max_i…
26                False  {'ls__alpha': 0.001, 'ls__max_iter': 5000, 'ls…
27                False  {'ls__alpha': 0.06199705881601676, 'ls__max_it…
28                 True  {'ls__alpha': 0.01687341186753009, 'ls__max_it…
29                False  {'ls__alpha': 0.0018185560271089783, 'ls__max_…
30                 True  {'ls__alpha': 0.0018303658598422532, 'ls__max_…
31                 True  {'ls__alpha': 0.013661350637267831, 'ls__max_i…
32                 True  {'ls__alpha': 0.04636797353331355, 'ls__max_it…
33                 True  {'ls__alpha': 0.006918445469792199, 'ls__max_i…
34                False  {'ls__alpha': 0.001, 'ls__max_iter': 100, 'ls_…
35                False  {'ls__alpha': 0.00421690504388654, 'ls__max_it…
36                 True  {'ls__alpha': 0.001, 'ls__max_iter': 5000, 'ls…
37                 True  {'ls__alpha': 0.010632072820022796, 'ls__max_i…
38                False  {'ls__alpha': 0.015438722672047024, 'ls__max_i…
39                False  {'ls__alpha': 0.0022740964287695036, 'ls__max_…
40                False  {'ls__alpha': 0.0022923216227139103, 'ls__max_…
41                 True  {'ls__alpha': 0.03324342462416447, 'ls__max_it…
42                False  {'ls__alpha': 0.009614323672543546, 'ls__max_i…
43                 True  {'ls__alpha': 0.02358308086767287, 'ls__max_it…
44                 True  {'ls__alpha': 0.08003292102153944, 'ls__max_it…
45                False  {'ls__alpha': 0.02600213461730265, 'ls__max_it…
```

```
46              False  {'ls__alpha': 0.0072884291262491525, 'ls__max_…
47              False  {'ls__alpha': 0.0011575416936216076, 'ls__max_…
48               True  {'ls__alpha': 0.08014202444546731, 'ls__max_it…
49              False  {'ls__alpha': 0.003942785721175991, 'ls__max_i…

    split0_test_score  split1_test_score  split2_test_score  \
0          -11.977457         -12.082645         -12.155722
1          -18.828273         -18.816528         -18.873667
2          -12.217014         -12.327770         -12.406181
3          -16.469237         -16.476761         -16.486357
4          -18.828273         -18.816528         -18.873667
5          -11.380662         -11.395944         -11.410907
6          -11.507037         -11.556216         -11.604021
7          -12.747390         -12.846645         -12.912869
8          -18.828273         -18.816528         -18.873667
9          -11.955538         -12.058478         -12.134445
10         -11.380589         -11.386755         -11.415462
11         -11.379845         -11.396099         -11.409756
12         -11.381898         -11.386847         -11.425501
13         -11.379818         -11.398849         -11.406317
14         -11.378792         -11.392354         -11.410356
15         -11.379706         -11.398501         -11.406496
16         -11.400669         -11.413166         -11.456496
17         -11.382560         -11.395759         -11.412853
18         -11.380164         -11.395063         -11.410263
19         -11.380880         -11.382676         -11.418229
20         -11.381084         -11.389948         -11.414542
21         -11.378914         -11.394251         -11.408826
22         -11.381145         -11.383385         -11.418390
23         -11.379440         -11.396892         -11.407253
24         -11.380317         -11.382732         -11.420022
25         -11.379224         -11.395523         -11.408173
26         -11.379918         -11.398930         -11.406477
27         -11.384259         -11.389523         -11.429869
28         -11.381018         -11.388484         -11.415529
29         -11.382439         -11.395379         -11.413013
30         -11.379603         -11.398171         -11.406777
31         -11.379134         -11.389215         -11.413041
32         -11.381865         -11.384254         -11.423136
33         -11.381857         -11.392092         -11.413386
34         -11.379834         -11.397019         -11.409298
35         -11.379235         -11.395867         -11.407826
36         -11.379818         -11.398849         -11.406317
37         -11.378559         -11.390977         -11.411400
38         -11.379298         -11.388353         -11.414127
39         -11.379691         -11.397716         -11.407038
40         -11.379802         -11.396319         -11.409414
41         -11.380290         -11.382304         -11.417945
```

```
42         -11.380949         -11.394112         -11.411515
43         -11.380230         -11.385291         -11.416149
44         -11.388175         -11.396595         -11.436773
45         -11.381719         -11.385534         -11.414868
46         -11.378868         -11.393370         -11.409555
47         -11.379802         -11.398752         -11.406384
48         -11.388123         -11.396783         -11.437692
49         -11.379968         -11.395684         -11.409814

    split3_test_score  split4_test_score  mean_test_score  std_test_score  \
0          -12.199795         -12.052402       -12.093604        0.078029
1          -18.934900         -19.055553       -18.901784        0.087412
2          -12.424359         -12.291528       -12.333370        0.076028
3          -16.600301         -16.742277       -16.554987        0.105179
4          -18.934900         -19.055553       -18.901784        0.087412
5          -11.504640         -11.387121       -11.415855        0.045537
6          -11.705059         -11.555625       -11.585592        0.067147
7          -12.945755         -12.854852       -12.861502        0.067858
8          -18.934900         -19.055553       -18.901784        0.087412
9          -12.179510         -12.029071       -12.071409        0.078797
10         -11.510673         -11.389846       -11.416665        0.048486
11         -11.503854         -11.386609       -11.415232        0.045434
12         -11.522605         -11.397110       -11.422792        0.052142
13         -11.500890         -11.385568       -11.414288        0.044305
14         -11.504997         -11.386338       -11.414567        0.046402
15         -11.501125         -11.385462       -11.414258        0.044446
16         -11.555529         -11.423823       -11.449937        0.055954
17         -11.506265         -11.388701       -11.417228        0.045658
18         -11.503716         -11.386209       -11.415083        0.045460
19         -11.513630         -11.391245       -11.417332        0.049969
20         -11.509086         -11.388563       -11.416645        0.047573
21         -11.503610         -11.385897       -11.414300        0.045754
22         -11.515097         -11.391125       -11.417828        0.050411
23         -11.502023         -11.385352       -11.414192        0.044947
24         -11.516632         -11.393011       -11.418543        0.051028
25         -11.502993         -11.385667       -11.414316        0.045402
26         -11.501023         -11.385705       -11.414411        0.044312
27         -11.526411         -11.399865       -11.425986        0.052637
28         -11.510211         -11.389163       -11.416881        0.048111
29         -11.506586         -11.388744       -11.417232        0.045829
30         -11.501183         -11.385397       -11.414226        0.044508
31         -11.507987         -11.387405       -11.415356        0.047671
32         -11.519297         -11.394784       -11.420667        0.051450
33         -11.508549         -11.388722       -11.416921        0.047009
34         -11.503119         -11.385764       -11.415007        0.045193
35         -11.502647         -11.385574       -11.414230        0.045258
36         -11.500890         -11.385568       -11.414288        0.044305
37         -11.505984         -11.386739       -11.414732        0.046892
```

```
38         -11.508911          -11.388004       -11.415739         0.048022
39         -11.501598          -11.385604       -11.414330         0.044654
40         -11.503101          -11.385844       -11.414896         0.045235
41         -11.514004          -11.391430       -11.417194         0.050231
42         -11.504158          -11.387072       -11.415561         0.045465
43         -11.511637          -11.390373       -11.416736         0.049037
44         -11.533929          -11.406318       -11.432358         0.053377
45         -11.511093          -11.389242       -11.416491         0.048708
46         -11.504270          -11.386175       -11.414448         0.046047
47         -11.500968          -11.385536       -11.414288         0.044347
48         -11.536468          -11.407262       -11.433265         0.054250
49         -11.503447          -11.385891       -11.414961         0.045382

    rank_test_score
0                44
1                48
2                45
3                47
4                48
5                23
6                42
7                46
8                48
9                43
10               26
11               19
12               37
13                5
14               13
15                4
16               41
17               31
18               18
19               33
20               25
21                8
22               34
23                1
24               35
25                9
26               11
27               38
28               28
29               32
30                2
31               20
32               36
33               29
```

```
34                17
35                 3
36                 5
37                14
38                22
39                10
40                15
41                30
42                21
43                27
44                39
45                24
46                12
47                 7
48                40
49                16
[ 0.80813214  2.15188504 20.47191416 …  4.87480063 20.68240508
  9.20470403]
[43.3663052   3.55764009 36.54084305 … -0.57203178  1.09980524
  3.14855382]
Train R-sq:
0.6370404443324977
Test R-sq:
0.6480368310543366
Train RMSE:
11.387518138430686
Test RMSE:
11.337839706231485
End Time = 2023-04-13 19:34:32.039668
Script Time = 0:04:34.328540
```

[15]:

```
coef_intercept = np.hstack((m2v2_ls.best_estimator_.named_steps["ls"].coef_,
                            m2v2_ls.best_estimator_.named_steps["ls"].
 ↪intercept_))
#print(coef_intercept)

coef_intercept_df01 = pd.DataFrame(coef_intercept)
#display(coef_intercept_df01)

train_x01_col_names = list(train_x01.columns)
train_x01_col_names.append('intercept')

train_x01_col_names_df01 = pd.DataFrame(train_x01_col_names)
#display(train_x01_col_names_df01)

model_params = pd.concat([train_x01_col_names_df01, coef_intercept_df01],␣
 ↪axis=1)
```

```
display(model_params)
```

```
                            0          0
0               borough_bronx  -0.059077
1            borough_brooklyn   1.203570
2           borough_manhattan   0.000000
3              borough_queens  -1.037298
4       borough_staten island   1.133191
5       relative_data_year_-4  -0.000000
6       relative_data_year_-3  -0.000000
7       relative_data_year_-2   0.002948
8       relative_data_year_-1   0.033875
9        relative_data_year_0  -0.000205
10      complaint_type_FELONY  -0.028230
11 complaint_type_MISDEMEANOR   0.020683
12   complaint_type_VIOLATION   0.000000
13 annual_evictions_x_borough  -0.000000
14    annual_complaint_counts  -0.043459
15              annual_grad_n   0.000000
16       annual_dropped_out_n  -0.000000
17                   totalpop   8.480980
18                        men   0.000000
19                      women   2.768329
20                   hispanic   0.697560
21                      white  -3.303184
22                      black  -2.148272
23                     native  -0.371075
24                      asian  -1.907961
25                    citizen  -3.655108
26                     income  -6.880255
27                  incomeerr   0.154667
28               incomepercap   0.443631
29            incomepercaperr  -0.176161
30               professional  -1.817674
31                    service   1.331155
32                     office   0.287453
33               construction  -0.798666
34                 production  -0.386056
35                      drive  -1.455973
36                    carpool  -0.486681
37                    transit   0.000000
38                       walk   1.527190
39                othertransp   0.485590
40                 workathome   0.568145
41                meancommute  -0.749279
42                   employed  -8.298285
43                privatework   0.000000
44                 publicwork  -1.698774
```

```
45                selfemployed   0.242544
46                  familywork   0.716689
47                unemployment   1.608683
48                   intercept  24.475273
```

[16]:

```
# specify the S3 bucket and key where you want to save the model
m2v2_ls_key_name = 'team_8_data/models/m2v2_ls.parquet'

# save the model to an in-memory buffer
buffer = BytesIO()
joblib.dump(m2v2_ls, buffer)

# upload the buffer to S3
buffer.seek(0)
s3.upload_fileobj(buffer, def_bucket, m2v2_ls_key_name)

# load the saved model from S3
#buffer = BytesIO()
#s3.download_fileobj(def_bucket, m2v2_ls_key_name, buffer)
#buffer.seek(0)
#m2v2_ls_fitted = joblib.load(buffer)
```

2.7 Release Resources

[17]:

```
%%html

<p><b>Shutting down your kernel for this notebook to release resources.</b></p>
<button class="sm-command-button" data-commandlinker-command="kernelmenu:
 ↪shutdown" style="display:none;">Shutdown Kernel</button>

<script>
try {
    els = document.getElementsByClassName("sm-command-button");
    els[0].click();
}
catch(err) {
    // NoOp
}
</script>
```

```
<IPython.core.display.HTML object>
```

[18]:

```
%%javascript

try {
    Jupyter.notebook.save_checkpoint();
    Jupyter.notebook.session.delete();
}
```

```
catch(err) {
    // NoOp
}
```

```
<IPython.core.display.Javascript object>
```