Diamonds

May 6, 2019

[1]:
```
import numpy as np
import pandas as pd
import pickle
import time
from scipy.stats import randint, uniform, describe
from sklearn import svm, preprocessing, linear_model, dummy, \
                    ensemble, neighbors, neural_network, tree
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.pipeline import make_pipeline, make_union
from tpot import TPOTRegressor
from tpot.builtins import StackingEstimator
from mlxtend.regressor import StackingRegressor
from pprint import pprint
from sklearn import metrics
from sklearn.base import BaseEstimator, RegressorMixin
%matplotlib inline
%config IPCompleter.greedy=True
```

```
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\deap\tools\_hypervolume\pyhv.py:33: ImportWarning: Falling back to the
python version of hypervolume module. Expect this to be very slow.
  "module. Expect this to be very slow.", ImportWarning)
c:\users\djesso\anaconda3\envs\python-
experiments\lib\importlib\_bootstrap.py:219: ImportWarning: can't resolve
package from __spec__ or __package__, falling back on __name__ and __path__
  return f(*args, **kwds)
```

[2]:
```
df = pd.read_csv("datasets/diamonds.csv", index_col=0)
df.head()
```

[2]:

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75

[3]:
```
# This is *super* useful if you category is arbitrary or order is unknown
# df["cut"].astype("category").cat.codes

# However, since we know the order, we put in a little work to encode them
cut_class_codes = {"Fair": 1, "Good": 2, "Very Good": 3, "Premium": 4, "Ideal":␣
 ↪5}
clarity_class_codes = {"I3": 1, "I2": 2, "I1": 3, "SI2": 4, "SI1": 5, "VS2": 6,␣
 ↪"VS1": 7, "VVS2": 8, "VVS1": 9, "IF": 10, "FL": 11}
color_class_codes = {"J": 1,"I": 2,"H": 3,"G": 4,"F": 5,"E": 6,"D": 7}

df["cut"] = df["cut"].map(cut_class_codes)
df["clarity"] = df["clarity"].map(clarity_class_codes)
df["color"] = df["color"].map(color_class_codes)
df.head()
```

[3]:

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	5	6	4	61.5	55.0	326	3.95	3.98	2.43
2	0.21	4	6	5	59.8	61.0	326	3.89	3.84	2.31
3	0.23	2	6	7	56.9	65.0	327	4.05	4.07	2.31
4	0.29	4	2	6	62.4	58.0	334	4.20	4.23	2.63
5	0.31	2	1	4	63.3	58.0	335	4.34	4.35	2.75

[4]:
```
df.describe()
```

[4]:

	carat	cut	color	clarity	depth	\
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000	
mean	0.797940	3.904097	4.405803	6.051020	61.749405	
std	0.474011	1.116600	1.701105	1.647136	1.432621	
min	0.200000	1.000000	1.000000	3.000000	43.000000	
25%	0.400000	3.000000	3.000000	5.000000	61.000000	
50%	0.700000	4.000000	4.000000	6.000000	61.800000	
75%	1.040000	5.000000	6.000000	7.000000	62.500000	
max	5.010000	5.000000	7.000000	10.000000	79.000000	

	table	price	x	y	z
count	53940.000000	53940.000000	53940.000000	53940.000000	53940.000000
mean	57.457184	3932.799722	5.731157	5.734526	3.538734
std	2.234491	3989.439738	1.121761	1.142135	0.705699
min	43.000000	326.000000	0.000000	0.000000	0.000000
25%	56.000000	950.000000	4.710000	4.720000	2.910000
50%	57.000000	2401.000000	5.700000	5.710000	3.530000
75%	59.000000	5324.250000	6.540000	6.540000	4.040000
max	95.000000	18823.000000	10.740000	58.900000	31.800000

Currently there's a minimum x y and z of 0, which isn't possible, so let's drop those rows and look at describe again

[5]:
```
df = df[(df[['x','y','z']] != 0).all(axis=1)]
df.describe()
```

[5]:
```
              carat           cut         color       clarity         depth  \
count  53920.000000  53920.000000  53920.000000  53920.000000  53920.000000
mean       0.797698      3.904228      4.405972      6.051502     61.749514
std        0.473795      1.116579      1.701272      1.647005      1.432331
min        0.200000      1.000000      1.000000      3.000000     43.000000
25%        0.400000      3.000000      3.000000      5.000000     61.000000
50%        0.700000      4.000000      4.000000      6.000000     61.800000
75%        1.040000      5.000000      6.000000      7.000000     62.500000
max        5.010000      5.000000      7.000000     10.000000     79.000000

              table         price             x             y             z
count  53920.000000  53920.000000  53920.000000  53920.000000  53920.000000
mean      57.456834   3930.993231      5.731627      5.734887      3.540046
std        2.234064   3987.280446      1.119423      1.140126      0.702530
min       43.000000    326.000000      3.730000      3.680000      1.070000
25%       56.000000    949.000000      4.710000      4.720000      2.910000
50%       57.000000   2401.000000      5.700000      5.710000      3.530000
75%       59.000000   5323.250000      6.540000      6.540000      4.040000
max       95.000000  18823.000000     10.740000     58.900000     31.800000
```

After dropping zeros, let's make some derived features

[6]:
```
df['volume'] = df['x']*df['y']*df['z']
df['density'] = df['carat'] / df['volume']
df['carat_table'] = df['carat'] * df['table']
df['carat_depth'] = df['carat'] * df['depth']
df['clarity_cut_color'] = df['clarity'] * df['cut'] * df['color']
df['carat_clarity_cut_color'] = df['carat'] * df['clarity_cut_color']
df.head()
```

[6]:
```
   carat  cut  color  clarity  depth  table  price     x     y     z  \
1   0.23    5      6        4   61.5   55.0    326  3.95  3.98  2.43
2   0.21    4      6        5   59.8   61.0    326  3.89  3.84  2.31
3   0.23    2      6        7   56.9   65.0    327  4.05  4.07  2.31
4   0.29    4      2        6   62.4   58.0    334  4.20  4.23  2.63
5   0.31    2      1        4   63.3   58.0    335  4.34  4.35  2.75

      volume   density  carat_table  carat_depth  clarity_cut_color  \
1  38.202030  0.006021        12.65       14.145                120
2  34.505856  0.006086        12.81       12.558                120
3  38.076885  0.006040        14.95       13.087                 84
4  46.724580  0.006207        16.82       18.096                 48
5  51.917250  0.005971        17.98       19.623                  8

   carat_clarity_cut_color
1                    27.60
2                    25.20
3                    19.32
4                    13.92
```

```
5                    2.48
```

[7]:
```
df.describe()
```

[7]:
```
              carat           cut         color       clarity         depth  \
count  53920.000000  53920.000000  53920.000000  53920.000000  53920.000000
mean       0.797698      3.904228      4.405972      6.051502     61.749514
std        0.473795      1.116579      1.701272      1.647005      1.432331
min        0.200000      1.000000      1.000000      3.000000     43.000000
25%        0.400000      3.000000      3.000000      5.000000     61.000000
50%        0.700000      4.000000      4.000000      6.000000     61.800000
75%        1.040000      5.000000      6.000000      7.000000     62.500000
max        5.010000      5.000000      7.000000     10.000000     79.000000

              table         price             x             y             z  \
count  53920.000000  53920.000000  53920.000000  53920.000000  53920.000000
mean      57.456834   3930.993231      5.731627      5.734887      3.540046
std        2.234064   3987.280446      1.119423      1.140126      0.702530
min       43.000000    326.000000      3.730000      3.680000      1.070000
25%       56.000000    949.000000      4.710000      4.720000      2.910000
50%       57.000000   2401.000000      5.700000      5.710000      3.530000
75%       59.000000   5323.250000      6.540000      6.540000      4.040000
max       95.000000  18823.000000     10.740000     58.900000     31.800000

             volume       density   carat_table   carat_depth  \
count  53920.000000  53920.000000  53920.000000  53920.000000
mean     129.897567      0.006127     46.025483     49.276657
std       78.219789      0.000178     27.744367     29.343722
min       31.707984      0.000521     11.000000     11.800000
25%       65.189759      0.006048     22.550000     24.520000
50%      114.840180      0.006117     40.320000     43.594000
75%      170.846415      0.006190     61.000000     64.872000
max     3840.598060      0.022647    295.590000    328.155000

       clarity_cut_color  carat_clarity_cut_color
count       53920.000000             53920.000000
mean          105.551298                72.580075
std            61.382226                49.310398
min             3.000000                 1.700000
25%            60.000000                37.260000
50%            96.000000                61.200000
75%           144.000000                94.400000
max           350.000000               409.500000
```

1 Feature importance

[8]:
```
lasso = linear_model.Lasso(alpha=0.0001)
features = df.drop(columns="price")
scaled_features = preprocessing.scale(features)
target = df["price"]
lasso.fit(scaled_features, target)

FI_lasso = pd.DataFrame({"Feature Importance": lasso.coef_},
                        index=features.columns)

FI_lasso.sort_values("Feature Importance", ascending=False)
importance = FI_lasso[FI_lasso["Feature Importance"] != 0].sort_values(
    "Feature Importance")
print(importance)
try:
    importance.plot(kind="barh", figsize=(25, 35))
    plt.xticks(rotation=90)
    plt.plot()
except:
    pass

print('[', end='')
for i in importance.index:
    print(f'\'{i}\'', end=',')
print(']', end='')
```

```
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\ipykernel_launcher.py:3: DataConversionWarning: Data with input dtype
int64, float64 were all converted to float64 by the scale function.
  This is separate from the ipykernel package so we can avoid doing imports
until
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:492: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Fitting data with very small alpha may cause precision problems.
  ConvergenceWarning)
```

	Feature Importance
x	-2759.110468
clarity_cut_color	-917.439737
carat_depth	-499.569926
y	-405.773400
z	-241.202641
cut	-236.358452
table	-174.657900
depth	-150.369281
density	-84.356368

```
color                              6.781231
volume                           440.914746
clarity                          460.042407
carat_clarity_cut_color         1913.270318
carat_table                     2027.520975
carat                           4044.535181
['x','clarity_cut_color','carat_depth','y','z','cut','table','depth','density','
```

[9]:
```
features = df.drop("price", axis=1).values
features = preprocessing.scale(features)
target = df["price"].values
train_features, test_features, train_target, test_target =␣
↪train_test_split(features, target, test_size=0.25, shuffle=True)
```

2 Classes and helpers

[10]:
```
def negative_rmse(y_true, y_pred):
    return -(metrics.mean_squared_error(y_true, y_pred) ** (1 / 2))
```

[11]:
```
rmse_scorer = metrics.make_scorer(negative_rmse, greater_is_better=True)
```

[12]:
```
class AveragedRegressor(BaseEstimator, RegressorMixin):
    """Look, a regressor"""


    def __init__(self, metric=negative_rmse, bigger_is_better=True,␣
↪simple_average=False):
        """
        Called when initializing the classifier.
        Is literally empty because everything necessary is already above this␣
↪class.
        """
        self.models = []
        self.trained = False
        self.metric = metric
        self.score_min = None
        self.score_max = None
        self.bigger_is_better = bigger_is_better
        self.simple_average=simple_average



    def append(self, model):
        """
```

```
        Adds a model to the list of models.
        """
        self.models.append([0, model])



    def fit(self, X, y=None):
        """
        This should fit all the regressors and score them.
        All the "work" should be done here.
        """

        assert (len(self.models) > 0), "models must be a list of models to␣
↪train"

        train_features, test_features, train_target, test_target =␣
↪train_test_split(X, y, test_size=0.2, shuffle=True)

        for i in self.models:
            i[1].fit(train_features, train_target)
            if self.bigger_is_better:
                i[0] = self.metric(test_target, i[1].predict(test_features))
            else:
                i[0] = 1 / (self.metric(test_target, i[1].
↪predict(test_features)))

        self.models = sorted(self.models, reverse=True)

        scores = []
        for i in self.models:
            scores.append(i[0])

        self.score_min = np.min(scores)
        self.score_max = np.max(scores)
        if not self.simple_average:
            for i in self.models:
                i[0] = ((i[0] - self.score_min) / (self.score_max - self.
↪score_min))
        else:
            for i in self.models:
                i[0] = 1

        self.trained = True

        return self
```

```
    def predict(self, X):
        """
        Predicts data from trained models
        """
        if not self.trained:
            raise RuntimeError("You must train classifer before predicting␣
 ↪data")

        guess = 0
        divisor = 0
        for score, model in self.models:
            prediction = model.predict(X)
            guess += (prediction * score)
#             pprint(f"model {model} \nwith score {score} \nmade prediction␣
 ↪{prediction}")
            divisor += score
        guess /= divisor
        return guess


    def score(self, X, y=None):
        y_true = y
        y_pred = self.predict(X)
        pprint(f"y_true: {y_true}")
        pprint(f"y_pred: {y_pred}")
        print("\n")

        return self.metric(y_true, y_pred)


    def load(self, filename):
        f = open(filename, 'rb')
        tmp_dict = pickle.load(f)
        f.close()

        self.__dict__.update(tmp_dict)


    def save(self, filename):
        f = open(filename, 'wb')
        pickle.dump(self.__dict__, f, 2)
        f.close()
```

[13]:
```
class SingleBestRegressor(BaseEstimator, RegressorMixin):
    """Look, another regressor"""
```

```
    def __init__(self, metric=negative_rmse, bigger_is_better=True):
        """
        Called when initializing the classifier.
        Is literally empty because everything necessary is already above this␣
↪class.
        """
        self.models = []
        self.best_model = None
        self.trained = False
        self.metric = metric
        self.score_min = None
        self.score_max = None
        self.bigger_is_better = bigger_is_better



    def append(self, model):
        """
        Adds a model to the list of models.
        """
        self.models.append([0, model])



    def fit(self, X, y=None):
        """
        This should fit all the regressors and score them.
        All the "work" should be done here.
        """

        assert (len(self.models) > 0), "models must be a list of models to␣
↪train"

        train_features, test_features, train_target, test_target =␣
↪train_test_split(X, y, test_size=0.2, shuffle=True)

        for i in self.models:
            i[1].fit(train_features, train_target)
            if self.bigger_is_better:
                i[0] = self.metric(test_target, i[1].predict(test_features))
            else:
                i[0] = 1 / (self.metric(test_target, i[1].
↪predict(test_features)))

        self.models = sorted(self.models, reverse=True)
        self.best_model = self.models[0][1]
        del self.models
```

```
        self.trained = True

        return self


    def predict(self, X):
        """
        Predicts data from trained models
        """
        if not self.trained:
            raise RuntimeError("You must train classifer before predicting␣
↪data")

        return self.best_model.predict(X)


    def score(self, X, y=None):
        y_true = y
        y_pred = self.predict(X)
        pprint(f"y_true: {y_true}")
        pprint(f"y_pred: {y_pred}")
        print("\n")

        return self.metric(y_true, y_pred)


    def load(self, filename):
        f = open(filename, 'rb')
        tmp_dict = pickle.load(f)
        f.close()

        self.__dict__.update(tmp_dict)


    def save(self, filename):
        f = open(filename, 'wb')
        pickle.dump(self.__dict__, f, 2)
        f.close()
```

[14]:
```
def learn(func, show_results=False, **kwargs):
    start = time.time()
    model = func(**kwargs)
    model.fit(train_features, train_target)
    print(f"Score: {model.score(test_features, test_target)}")
    finish = time.time()
    print(f"Time: {finish-start}")
```

```
    if show_results:
        for i, actual in zip(test_features, test_target):
            guess = model.predict([i])[0]
            diff = guess - actual
            print(f"Difference: {diff:9.2f}\t\tModel: {guess:10.2f},\t\tActual:␣
 ↪{actual:10.0f}")
```

[15]:
```
# Utility function to report best scores
def report(results, n_top=3):
    for i in range(1, n_top + 1):
        candidates = np.flatnonzero(results['rank_test_score'] == i)
        for candidate in candidates:
            print("Model with rank: {0}".format(i))
            print("Mean validation score: {0:.3f} (std: {1:.3f})".format(
                  results['mean_test_score'][candidate],
                  results['std_test_score'][candidate]))
            print("Parameters: {0}\n".format(results['params'][candidate]))
```

[16]:
```
def n_best(results, n_top=5):
    models = []
    for i in range(1, n_top + 1):
        candidates = np.flatnonzero(results['rank_test_score'] == i)
```

[17]:
```
def randfloat(low=0, high=1):
    return uniform(low, high-low)
```

[18]:
```
def search_time(func, params, iterations=10):
#     iterations //= 50
    start = time.time()
    random_search = search(func, params, iterations)
    random_search.fit(features, target)
    report(random_search.cv_results_)
    finish = time.time()
    total = finish - start
    print(f"Took {total:.2f} seconds overall, ~{total/iterations:.2f} seconds␣
 ↪per fit")
    return random_search
```

[19]:
```
def search(func, params, iterations=10):
#     iterations //= 100
#     iterations += 1
#     iterations *= 10
    model = func()
    random_search = RandomizedSearchCV(model, param_distributions=params,
                                       n_iter=iterations,
                                       cv=5,
                                       scoring=rmse_scorer,
#                                         scoring="r2",
```

```
                                        n_jobs=-1,
                                        verbose=2
                                       )
    return random_search
```

[20]:
```
all_models = []
```

3 Dummy

[21]:
```
params = {"strategy": ["mean", "median"]}
search_time(dummy.DummyRegressor, params, iterations=2).fit(features, target)
```

```
Fitting 5 folds for each of 2 candidates, totalling 10 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of  10 | elapsed:    0.8s remaining:    0.8s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    0.8s finished

Model with rank: 1
Mean validation score: -3794.182 (std: 2177.587)
Parameters: {'strategy': 'mean'}

Model with rank: 2
Mean validation score: -3852.441 (std: 2443.208)
Parameters: {'strategy': 'median'}

Took 0.90 seconds overall, ~0.45 seconds per fit
Fitting 5 folds for each of 2 candidates, totalling 10 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   5 out of  10 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    0.2s finished
```

[21]:
```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
          estimator=DummyRegressor(constant=None, quantile=None,
strategy='mean'),
          fit_params=None, iid='warn', n_iter=2, n_jobs=-1,
          param_distributions={'strategy': ['mean', 'median']},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score='warn', scoring=make_scorer(negative_rmse),
          verbose=2)
```

4 Ensembles

[22]:
```
params = {
          "n_estimators": randint(10, 250),
          "max_features": ["sqrt", "log2", "auto"],
          "max_depth": randint(1, 15),
          "min_samples_split": randint(2, 100),
          "min_samples_leaf": randint(1, 100),
          "bootstrap": [True, False]
         }
all_models.append(search(ensemble.ExtraTreesRegressor, params, iterations=48))
# search_time(ensemble.ExtraTreesRegressor, params, iterations=48)
```

[23]:
```
params = {
          "loss": ["ls", "huber", "lad"],
          "learning_rate": randfloat(0.001, 0.2),
          "n_estimators": randint(10, 250),
          "subsample": randfloat(),
          "max_depth": randint(1, 15),
          "max_features": ["sqrt", "log2", "auto"],
         }

all_models.append(search(ensemble.GradientBoostingRegressor, params,␣
 ↪iterations=4))
```

[24]:
```
params = {
          "n_estimators": randint(10, 250),
          "min_samples_split": randint(2, 100),
          "min_samples_leaf": randint(1, 100),
          "max_features": ["auto", "sqrt", "log2"],
          "bootstrap": [True, False],
         }
all_models.append(search(ensemble.RandomForestRegressor, params, iterations=24))
```

5 Linear Models

[25]:
```
params = {
          "n_iter": randint(1, 1000),
          "tol": randfloat(),
          "alpha_1": randfloat(),
          "alpha_2": randfloat(),
          "lambda_1": randfloat(),
          "lambda_2": randfloat(),
         }
all_models.append(search(linear_model.BayesianRidge, params, iterations=1000))
```

[26]:
```
params = {
          "alpha": randfloat(),
          "l1_ratio": randfloat(),
          "tol": randfloat(),
          "positive": [True, False],
          "selection": ["cyclic", "random"],
         }
all_models.append(search(linear_model.ElasticNet, params, iterations=1500))
```

[27]:
```
params = {
          "epsilon": randfloat(1, 2),
          "max_iter": randint(50, 150),
          "alpha": randfloat(),
          "tol": randfloat(),
         }
all_models.append(search(linear_model.HuberRegressor, params, iterations=160))
```

[28]:
```
params = {
          "alpha": randfloat(),
          "tol": randfloat(),
          "positive": [True, False],
          "selection": ["cyclic", "random"],
         }

all_models.append(search(linear_model.Lasso, params, iterations=1500))
```

[29]:
```
params = {
          "eps": randfloat(),
         }

all_models.append(search(linear_model.Lars, params, iterations=2000))
```

[30]:
```
params = {
          "alpha": randfloat(),
          "eps": randfloat()
         }

all_models.append(search(linear_model.LassoLars, params, iterations=2000))
```

[31]:
```
all_models.append(search(linear_model.LinearRegression, {}, iterations=1))
```

[32]:
```
params = {
          "C": randfloat(0.0, 5)
         }

all_models.append(search(linear_model.PassiveAggressiveRegressor, params,␣
 ↪iterations=1500))
```

[33]:
```
params = {
          "min_samples": randfloat(),
```

```
            "residual_threshold": randfloat(),
            "stop_probability": randfloat(),
           }

all_models.append(search(linear_model.RANSACRegressor, params, iterations=50))
```

[34]:
```
params = {
            "alpha": randfloat(),
            "tol": randfloat(),
            "solver": ["svd", "cholesky", "lsqr", "sparse_cg", "sag", "saga"],
           }

all_models.append(search(linear_model.Ridge, params, iterations=2000))
```

[35]:
```
params = {
            "loss": ["squared_loss", "huber", "epsilon_insensitive",␣
 ↪"squared_epsilon_insensitive"],
            "penalty": ["none", "l2", "l1", "elasticnet"],
            "learning_rate": ["constant", "optimal", "invscaling", "adaptive"],
            "eta0": randfloat(),
            "power_t": randfloat(),
           }

all_models.append(search(linear_model.SGDRegressor, params, iterations=2000))
```

6 Nearest Neighbors

[36]:
```
params = {
            "n_neighbors": randint(1, 10),
            "weights": ["uniform", "distance"],
            "algorithm": ["ball_tree", "kd_tree", "brute"],
            "leaf_size": randint(1, 50),
            "p": randint(1, 3),
           }

all_models.append(search(neighbors.KNeighborsRegressor, params, iterations=3))
```

[37]:
```
params = {
            "radius": randfloat(0, 2),
            "weights": ["uniform", "distance"],
            "algorithm": ["ball_tree", "kd_tree", "brute"],
            "leaf_size": randint(1, 50),
            "p": randint(1, 3),
           }
```

```
# all_models.append(search(neighbors.RadiusNeighborsRegressor, params,␣
↪iterations=3))
```

7 Support Vector Machines

[38]:
```
params = {
          "epsilon": randfloat(),
          "tol": randfloat(),
          "C": randfloat(0, 2),
          "max_iter": randint(500, 1500)
         }

all_models.append(search(svm.LinearSVR, params, iterations=1500))
```

[39]:
```
# params = {
#           "nu": randfloat(),
#           "C": randfloat(0, 2),
#           "kernel": ["linear"],
# #           "kernel": ["linear", "poly", "rbf", "sigmoid"],
#          }

# all_models.append(search(svm.NuSVR, params, iterations=10))
```

[40]:
```
# params = {
#           "kernel": ["linear"],
# #           "kernel": ["linear", "poly", "rbf", "sigmoid"],
#           "C": randfloat(0, 2),
#          }

# all_models.append(search(svm.SVR, params, iterations=1))
```

8 Decision Trees

[41]:
```
params = {
          "splitter": ["best", "random"],
          "max_features": ["sqrt", "log2", "auto"],
          "min_samples_split": randfloat(),
         }

all_models.append(search(tree.DecisionTreeRegressor, params, iterations=2000))
```

[42]:
```
params = {
          "splitter": ["best", "random"],
          "max_features": ["sqrt", "log2", "auto"],
          "min_samples_split": randfloat(),
         }
```

```
all_models.append(search(tree.ExtraTreeRegressor, params, iterations=2000))
```

9 Setup for TPOT

[43]:
```
config_dict = {
    # Ensembles
    "sklearn.ensemble.AdaBoostRegressor": {
        "n_estimators": [100],
        "learning_rate": [1e-3, 1e-2, 1e-1, 0.5, 1.0],
        "loss": ["linear", "square", "exponential"],
    },
    "sklearn.ensemble.ExtraTreesRegressor": {
        "n_estimators": [100],
        "max_features": np.arange(0.05, 1.01, 0.05),
        "min_samples_split": range(2, 21),
        "min_samples_leaf": range(1, 21),
        "bootstrap": [True, False],
    },
    "sklearn.ensemble.GradientBoostingRegressor": {
        "n_estimators": [100],
        "loss": ["ls", "lad", "huber", "quantile"],
        "learning_rate": [1e-3, 1e-2, 1e-1, 0.5, 1.0],
        "max_depth": range(1, 11),
        "min_samples_split": range(2, 21),
        "min_samples_leaf": range(1, 21),
        "subsample": np.arange(0.05, 1.01, 0.05),
        "max_features": np.arange(0.05, 1.01, 0.05),
        "alpha": [0.75, 0.8, 0.85, 0.9, 0.95, 0.99],
    },
    "sklearn.ensemble.RandomForestRegressor": {
        "n_estimators": [100],
        "max_features": np.arange(0.05, 1.01, 0.05),
        "min_samples_split": range(2, 21),
        "min_samples_leaf": range(1, 21),
        "bootstrap": [True, False],
    },
    # Linear Models
    "sklearn.linear_model.ElasticNetCV": {
        "l1_ratio": np.arange(0.0, 1.01, 0.05),
        "tol": [1e-5, 1e-4, 1e-3, 1e-2, 1e-1],
    },
    "sklearn.linear_model.LassoLarsCV": {"normalize": [True, False]},
    "sklearn.linear_model.RidgeCV": {
        "alpha": np.arange(0.0, 1.01, 0.05),
        "tol": np.arange(0.0, 1.01, 0.05),
```

```
        "solver": ["svd", "cholesky", "lsqr", "sparse_cg", "sag", "saga"],
    },
    "sklearn.linear_model.SGDRegressor": {
        "loss": [
            "squared_loss",
            "huber",
            "epsilon_insensitive",
            "squared_epsilon_insensitive",
        ],
        "penalty": ["none", "l2", "l1", "elasticnet"],
        "learning_rate": ["constant", "optimal", "invscaling", "adaptive"],
        "eta0": np.arange(0.0, 1.01, 0.05),
        "power_t": np.arange(0.0, 1.01, 0.05),
    },
    # Neighbors
    "sklearn.neighbors.KNeighborsRegressor": {
        "n_neighbors": range(1, 101),
        "weights": ["uniform", "distance"],
        "p": [1, 2],
    },
    # SVM
    "sklearn.svm.LinearSVR": {
        "loss": ["epsilon_insensitive", "squared_epsilon_insensitive"],
        "dual": [True, False],
        "tol": [1e-5, 1e-4, 1e-3, 1e-2, 1e-1],
        "C": [1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1.0, 5.0, 10.0, 15.0, 20.0, 25.0],
        "epsilon": [1e-4, 1e-3, 1e-2, 1e-1, 1.0],
    },
    # Tree
    "sklearn.tree.DecisionTreeRegressor": {
        "max_depth": range(1, 11),
        "min_samples_split": range(2, 21),
        "min_samples_leaf": range(1, 21),
    },
    # xgboost
    "xgboost.XGBRegressor": {
        "n_estimators": [100],
        "max_depth": range(1, 11),
        "learning_rate": [1e-3, 1e-2, 1e-1, 0.5, 1.0],
        "subsample": np.arange(0.05, 1.01, 0.05),
        "min_child_weight": range(1, 21),
        "nthread": [1],
    },
    # Preprocesssors
    "sklearn.preprocessing.Binarizer": {"threshold": np.arange(0.0, 1.01, 0.
↪05)},
    "sklearn.decomposition.FastICA": {"tol": np.arange(0.0, 1.01, 0.05)},
```

```
    "sklearn.cluster.FeatureAgglomeration": {
        "linkage": ["ward", "complete", "average"],
        "affinity": ["euclidean", "l1", "l2", "manhattan", "cosine"],
    },
    "sklearn.preprocessing.MaxAbsScaler": {},
    "sklearn.preprocessing.MinMaxScaler": {},
    "sklearn.preprocessing.Normalizer": {"norm": ["l1", "l2", "max"]},
    "sklearn.kernel_approximation.Nystroem": {
        "kernel": [
            "rbf",
            "cosine",
            "chi2",
            "laplacian",
            "polynomial",
            "poly",
            "linear",
            "additive_chi2",
            "sigmoid",
        ],
        "gamma": np.arange(0.0, 1.01, 0.05),
        "n_components": range(1, 11),
    },
    "sklearn.decomposition.PCA": {
        "svd_solver": ["randomized"],
        "iterated_power": range(1, 11),
    },
    "sklearn.preprocessing.PolynomialFeatures": {
        "degree": [2],
        "include_bias": [False],
        "interaction_only": [False],
    },
    "sklearn.kernel_approximation.RBFSampler": {"gamma": np.arange(0.0, 1.01, 0.
↪05)},
    "sklearn.preprocessing.RobustScaler": {},
    "sklearn.preprocessing.StandardScaler": {},
    "tpot.builtins.ZeroCount": {},
    "tpot.builtins.OneHotEncoder": {
        "minimum_fraction": [0.05, 0.1, 0.15, 0.2, 0.25],
        "sparse": [False],
        "threshold": [10],
    },
    # Selectors
    "sklearn.feature_selection.SelectFwe": {
        "alpha": np.arange(0, 0.05, 0.001),
        "score_func": {"sklearn.feature_selection.f_regression": None},
    },
    "sklearn.feature_selection.SelectPercentile": {
```

```
        "percentile": range(1, 100),
        "score_func": {"sklearn.feature_selection.f_regression": None},
    },
    "sklearn.feature_selection.VarianceThreshold": {
        "threshold": [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2]
    },
    "sklearn.feature_selection.SelectFromModel": {
        "threshold": np.arange(0, 1.01, 0.05),
        "estimator": {
            "sklearn.ensemble.ExtraTreesRegressor": {
                "n_estimators": [100],
                "max_features": np.arange(0.05, 1.01, 0.05),
            }
        },
    },
}
```

[44]:
```
weighted_average_regressors = AveragedRegressor()
simple_average_regressors = AveragedRegressor(simple_average=True)
best_single_regressor = SingleBestRegressor()
for i in all_models:
    weighted_average_regressors.append(i)
    simple_average_regressors.append(i)
    best_single_regressor.append(i)
```

[45]:
```
weighted_average_regressors.fit(train_features, train_target)
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   12.2s
[Parallel(n_jobs=-1)]: Done 138 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 240 out of 240 | elapsed:  1.7min finished

Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 out of  20 | elapsed:    1.1s remaining:    1.8s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    6.4s finished

Fitting 5 folds for each of 24 candidates, totalling 120 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   11.1s
[Parallel(n_jobs=-1)]: Done 120 out of 120 | elapsed:  1.4min finished

Fitting 5 folds for each of 1000 candidates, totalling 5000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.4s
```

```
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    5.2s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:   13.2s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   24.4s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   38.8s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   56.2s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 5000 out of 5000 | elapsed:  1.6min finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    3.3s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    7.7s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   15.7s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   24.0s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   35.3s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   47.0s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.5min finished
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\sklearn\linear_model\coordinate_descent.py:492: ConvergenceWarning:
Objective did not converge. You might want to increase the number of iterations.
Fitting data with very small alpha may cause precision problems.
  ConvergenceWarning)

Fitting 5 folds for each of 160 candidates, totalling 800 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    3.4s
[Parallel(n_jobs=-1)]: Done 138 tasks      | elapsed:   23.9s
[Parallel(n_jobs=-1)]: Done 341 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 624 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done 800 out of 800 | elapsed:  2.3min finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    7.5s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   16.3s
[Parallel(n_jobs=-1)]: Done 1666 tasks      | elapsed:   25.0s
[Parallel(n_jobs=-1)]: Done 2114 tasks      | elapsed:   31.0s
[Parallel(n_jobs=-1)]: Done 2641 tasks      | elapsed:   37.0s
[Parallel(n_jobs=-1)]: Done 3248 tasks      | elapsed:   45.8s
[Parallel(n_jobs=-1)]: Done 3937 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 4706 tasks      | elapsed:  1.2min
```

```
[Parallel(n_jobs=-1)]: Done 5557 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 6488 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.9min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    3.8s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   10.5s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   19.5s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   31.3s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   45.7s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.3min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   10.5s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   19.6s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   31.3s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   45.8s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.3min finished

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

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of   5 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    0.0s finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.3s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    6.1s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   11.3s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   18.0s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   26.2s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   36.1s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   47.2s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:   59.9s
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.2min finished
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:
```

```
max_iter and tol parameters have been added in PassiveAggressiveRegressor in
0.19. If both are left unset, they default to max_iter=5 and tol=None. If tol is
not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will
be 1000, and default tol will be 1e-3.
  FutureWarning)

Fitting 5 folds for each of 50 candidates, totalling 250 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   13.1s
[Parallel(n_jobs=-1)]: Done 138 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 250 out of 250 | elapsed:  1.8min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.7s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    7.1s
[Parallel(n_jobs=-1)]: Done 1248 tasks      | elapsed:   13.5s
[Parallel(n_jobs=-1)]: Done 2314 tasks      | elapsed:   25.4s
[Parallel(n_jobs=-1)]: Done 3204 tasks      | elapsed:   34.9s
[Parallel(n_jobs=-1)]: Done 4260 tasks      | elapsed:   46.7s
[Parallel(n_jobs=-1)]: Done 5000 tasks      | elapsed:   56.2s
[Parallel(n_jobs=-1)]: Done 5964 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 7386 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 9088 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  2.1min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    4.4s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   11.7s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   21.9s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   35.1s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   51.2s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.5min finished
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:
max_iter and tol parameters have been added in SGDRegressor in 0.19. If both are
left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000,
and default tol will be 1e-3.
  FutureWarning)

Fitting 5 folds for each of 3 candidates, totalling 15 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 out of  15 | elapsed:  1.5min remaining:  1.3min
[Parallel(n_jobs=-1)]: Done  15 out of  15 | elapsed:  1.6min finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    7.4s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   13.9s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   22.1s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   32.5s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   44.6s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   58.3s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.4min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    4.1s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   10.7s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   20.2s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   32.3s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   46.8s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.4min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   10.6s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   20.1s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   31.9s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   46.6s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.4min finished
```

[45]:
```
AveragedRegressor(bigger_is_better=True,
         metric=<function negative_rmse at 0x0000016F972B7048>,
         simple_average=False)
```

[46]:
```
simple_average_regressors.fit(train_features, train_target)
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done 138 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 240 out of 240 | elapsed:  1.7min finished

Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 out of  20 | elapsed:    7.1s remaining:   10.7s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:   37.1s finished

Fitting 5 folds for each of 24 candidates, totalling 120 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    6.1s
[Parallel(n_jobs=-1)]: Done 120 out of 120 | elapsed:  1.0min finished

Fitting 5 folds for each of 1000 candidates, totalling 5000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    5.1s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:   12.8s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   23.5s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   37.6s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   54.8s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 5000 out of 5000 | elapsed:  1.6min finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    3.5s
[Parallel(n_jobs=-1)]: Done 639 tasks      | elapsed:   10.6s
[Parallel(n_jobs=-1)]: Done 924 tasks      | elapsed:   14.2s
[Parallel(n_jobs=-1)]: Done 1289 tasks      | elapsed:   19.2s
[Parallel(n_jobs=-1)]: Done 1736 tasks      | elapsed:   24.6s
[Parallel(n_jobs=-1)]: Done 2263 tasks      | elapsed:   31.4s
[Parallel(n_jobs=-1)]: Done 2870 tasks      | elapsed:   39.4s
[Parallel(n_jobs=-1)]: Done 3559 tasks      | elapsed:   48.4s
[Parallel(n_jobs=-1)]: Done 4328 tasks      | elapsed:   58.9s
[Parallel(n_jobs=-1)]: Done 5179 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 6110 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 7123 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.6min finished

Fitting 5 folds for each of 160 candidates, totalling 800 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    3.3s
[Parallel(n_jobs=-1)]: Done 138 tasks      | elapsed:   22.4s
[Parallel(n_jobs=-1)]: Done 341 tasks      | elapsed:   55.4s
[Parallel(n_jobs=-1)]: Done 624 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 800 out of 800 | elapsed:  2.2min finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    7.2s
[Parallel(n_jobs=-1)]: Done 1146 tasks      | elapsed:   15.3s
[Parallel(n_jobs=-1)]: Done 1514 tasks      | elapsed:   21.6s
[Parallel(n_jobs=-1)]: Done 1962 tasks      | elapsed:   28.6s
[Parallel(n_jobs=-1)]: Done 2489 tasks      | elapsed:   36.4s
[Parallel(n_jobs=-1)]: Done 3096 tasks      | elapsed:   44.3s
[Parallel(n_jobs=-1)]: Done 3785 tasks      | elapsed:   53.4s
[Parallel(n_jobs=-1)]: Done 4554 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 5405 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 6336 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 7349 tasks      | elapsed:  1.6min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.7min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.1s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    5.5s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   10.2s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   16.3s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   23.8s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   32.7s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   42.8s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:   54.3s
[Parallel(n_jobs=-1)]: Done 8028 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 9730 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.4min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.2s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    5.7s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   10.6s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   16.9s
```

```
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   24.7s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   33.8s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   44.5s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:   56.6s
[Parallel(n_jobs=-1)]: Done 8028 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 9730 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.5min finished

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

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of   5 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    0.0s finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.4s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    6.1s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   11.4s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   18.3s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   26.6s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   36.3s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   47.6s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.2min finished
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:
max_iter and tol parameters have been added in PassiveAggressiveRegressor in
0.19. If both are left unset, they default to max_iter=5 and tol=None. If tol is
not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will
be 1000, and default tol will be 1e-3.
  FutureWarning)

Fitting 5 folds for each of 50 candidates, totalling 250 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    7.3s
[Parallel(n_jobs=-1)]: Done 138 tasks      | elapsed:   57.6s
[Parallel(n_jobs=-1)]: Done 250 out of 250 | elapsed:  1.9min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.7s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    6.9s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   13.2s
```

```
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   21.7s
[Parallel(n_jobs=-1)]: Done 2693 tasks      | elapsed:   30.2s
[Parallel(n_jobs=-1)]: Done 3734 tasks      | elapsed:   41.6s
[Parallel(n_jobs=-1)]: Done 4749 tasks      | elapsed:   53.1s
[Parallel(n_jobs=-1)]: Done 5943 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 7479 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 9181 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 9977 out of 10000 | elapsed:  1.9min remaining:
0.2s
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  2.0min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.3s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    6.1s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   11.4s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   18.1s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   26.4s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   36.2s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   47.4s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 8028 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 9730 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.5min finished
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:
max_iter and tol parameters have been added in SGDRegressor in 0.19. If both are
left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000,
and default tol will be 1e-3.
  FutureWarning)

Fitting 5 folds for each of 3 candidates, totalling 15 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 out of  15 | elapsed:  2.2min remaining:  1.9min
[Parallel(n_jobs=-1)]: Done  15 out of  15 | elapsed:  2.3min finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    7.1s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   13.2s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   21.2s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   31.2s
```

```
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   43.0s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   56.6s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.4min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    4.2s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   10.8s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   20.2s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   32.3s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   47.1s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.4min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   10.8s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   20.3s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   32.4s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   47.2s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.4min finished
```

[46]:
```
AveragedRegressor(bigger_is_better=True,
         metric=<function negative_rmse at 0x0000016F972B7048>,
         simple_average=True)
```

[47]:
```
best_single_regressor.fit(train_features, train_target)
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    9.6s
[Parallel(n_jobs=-1)]: Done 138 tasks      | elapsed:   51.6s
[Parallel(n_jobs=-1)]: Done 240 out of 240 | elapsed:  1.5min finished

Fitting 5 folds for each of 4 candidates, totalling 20 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 out of  20 | elapsed:    2.5s remaining:    3.8s
[Parallel(n_jobs=-1)]: Done  20 out of  20 | elapsed:    5.8s finished

Fitting 5 folds for each of 24 candidates, totalling 120 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   15.4s
[Parallel(n_jobs=-1)]: Done 120 out of 120 | elapsed:  1.9min finished

Fitting 5 folds for each of 1000 candidates, totalling 5000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.4s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    4.8s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:   12.5s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   23.5s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   37.5s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   54.8s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 5000 out of 5000 | elapsed:  1.6min finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.9s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    8.0s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   15.2s
[Parallel(n_jobs=-1)]: Done 1649 tasks      | elapsed:   20.6s
[Parallel(n_jobs=-1)]: Done 2094 tasks      | elapsed:   26.3s
[Parallel(n_jobs=-1)]: Done 2621 tasks      | elapsed:   33.2s
[Parallel(n_jobs=-1)]: Done 3228 tasks      | elapsed:   41.6s
[Parallel(n_jobs=-1)]: Done 3917 tasks      | elapsed:   50.2s
[Parallel(n_jobs=-1)]: Done 4686 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 5537 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 6468 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.6min finished

Fitting 5 folds for each of 160 candidates, totalling 800 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    4.0s
[Parallel(n_jobs=-1)]: Done 138 tasks      | elapsed:   23.7s
[Parallel(n_jobs=-1)]: Done 341 tasks      | elapsed:   58.7s
[Parallel(n_jobs=-1)]: Done 624 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done 800 out of 800 | elapsed:  2.3min finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    3.2s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    9.0s
[Parallel(n_jobs=-1)]: Done 984 tasks      | elapsed:   13.8s
```

```
[Parallel(n_jobs=-1)]: Done 1351 tasks      | elapsed:   22.6s
[Parallel(n_jobs=-1)]: Done 1798 tasks      | elapsed:   29.5s
[Parallel(n_jobs=-1)]: Done 2325 tasks      | elapsed:   41.2s
[Parallel(n_jobs=-1)]: Done 2932 tasks      | elapsed:   50.5s
[Parallel(n_jobs=-1)]: Done 3621 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 4390 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 5241 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 6172 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 7185 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  2.1min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    3.8s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   10.3s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   19.7s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   31.8s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   46.4s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.4min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.1s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.2s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    5.6s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   10.4s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   16.6s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   24.2s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   33.3s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   43.6s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:   55.4s
[Parallel(n_jobs=-1)]: Done 8028 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 9730 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.4min finished

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

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of   5 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    0.0s finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
```

```
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.4s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    6.0s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   11.4s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   18.3s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   26.5s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   36.3s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   47.5s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.2min finished
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:
max_iter and tol parameters have been added in PassiveAggressiveRegressor in
0.19. If both are left unset, they default to max_iter=5 and tol=None. If tol is
not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will
be 1000, and default tol will be 1e-3.
  FutureWarning)

Fitting 5 folds for each of 50 candidates, totalling 250 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:   12.3s
[Parallel(n_jobs=-1)]: Done 138 tasks      | elapsed:   57.0s
[Parallel(n_jobs=-1)]: Done 250 out of 250 | elapsed:  1.9min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.3s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    6.4s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   12.2s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   19.9s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   29.3s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   41.0s
[Parallel(n_jobs=-1)]: Done 4802 tasks      | elapsed:   50.7s
[Parallel(n_jobs=-1)]: Done 6180 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 7718 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 9420 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.8min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.5s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    6.2s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   11.5s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   18.4s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   26.7s
```

```
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   36.6s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   48.1s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 8028 tasks      | elapsed:  1.3min
[Parallel(n_jobs=-1)]: Done 9730 tasks      | elapsed:  1.5min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.6min finished
c:\users\djesso\anaconda3\envs\python-experiments\lib\site-
packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:
max_iter and tol parameters have been added in SGDRegressor in 0.19. If both are
left unset, they default to max_iter=5 and tol=None. If tol is not None,
max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000,
and default tol will be 1e-3.
  FutureWarning)

Fitting 5 folds for each of 3 candidates, totalling 15 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done   8 out of  15 | elapsed:   14.4s remaining:   12.6s
[Parallel(n_jobs=-1)]: Done  15 out of  15 | elapsed:   16.9s finished

Fitting 5 folds for each of 1500 candidates, totalling 7500 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 252 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 658 tasks      | elapsed:    7.2s
[Parallel(n_jobs=-1)]: Done 1224 tasks      | elapsed:   13.4s
[Parallel(n_jobs=-1)]: Done 1954 tasks      | elapsed:   21.4s
[Parallel(n_jobs=-1)]: Done 2844 tasks      | elapsed:   31.3s
[Parallel(n_jobs=-1)]: Done 3898 tasks      | elapsed:   43.3s
[Parallel(n_jobs=-1)]: Done 5112 tasks      | elapsed:   56.9s
[Parallel(n_jobs=-1)]: Done 6490 tasks      | elapsed:  1.2min
[Parallel(n_jobs=-1)]: Done 7500 out of 7500 | elapsed:  1.4min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    4.2s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   11.1s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   20.4s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   32.7s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   47.4s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.4min finished

Fitting 5 folds for each of 2000 candidates, totalling 10000 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  17 tasks      | elapsed:    0.2s
```

```
[Parallel(n_jobs=-1)]: Done 480 tasks      | elapsed:    4.1s
[Parallel(n_jobs=-1)]: Done 1292 tasks      | elapsed:   10.8s
[Parallel(n_jobs=-1)]: Done 2424 tasks      | elapsed:   20.5s
[Parallel(n_jobs=-1)]: Done 3884 tasks      | elapsed:   32.5s
[Parallel(n_jobs=-1)]: Done 5664 tasks      | elapsed:   47.5s
[Parallel(n_jobs=-1)]: Done 7772 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 10000 out of 10000 | elapsed:  1.4min finished
```

[47]:
```
SingleBestRegressor(bigger_is_better=True,
          metric=<function negative_rmse at 0x0000016F972B7048>)
```

[48]:
```
simple_average_regressors.score(test_features, test_target)
```

```
'y_true: [1141  770  696 ...  571  778  999]'
('y_pred: [1309.5635402   409.33626366  522.7758654  ...  133.48408627  '
 '868.14109466\n'
 '  837.21288656]')
```

[48]:
```
-726.3675103064977
```

[49]:
```
weighted_average_regressors.score(test_features, test_target)
```

```
'y_true: [1141  770  696 ...  571  778  999]'
('y_pred: [1332.10280425  511.93563139  596.47136552 ...  336.0316964   '
 '849.14351036\n'
 '  903.63981276]')
```

[49]:
```
-625.2447757706211
```

[50]:
```
best_single_regressor.score(test_features, test_target)
```

```
'y_true: [1141  770  696 ...  571  778  999]'
('y_pred: [1397.54735918  674.858574    713.2889548  ...  597.28969038  '
 '806.14806765\n'
 ' 1025.94845563]')
```

[50]:
```
-572.9178777392304
```

10 TPOT

[54]:
```
tpot = TPOTRegressor(generations=10, population_size=50, verbosity=3,␣
 ↪scoring=rmse_scorer, config_dict=config_dict, use_dask=True)
tpot.fit(train_features, train_target)

# tpot.export("best_fit.py")
```

```
30 operators have been imported by TPOT.

HBox(children=(IntProgress(value=0, description='Optimization Progress', max=550, style=Progres


_pre_test decorator: _random_mutation_operator: num_test=0 Found array with 0
feature(s) (shape=(50, 0)) while a minimum of 1 is required.
Pipeline encountered that has previously been evaluated during the optimization
process. Using the score from the previous evaluation.
Pipeline encountered that has previously been evaluated during the optimization
process. Using the score from the previous evaluation.
Generation 1 - Current Pareto front scores:
-1      -550.3739539547902      ExtraTreesRegressor(input_matrix,
ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.8500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=13,
ExtraTreesRegressor__n_estimators=100)

_pre_test decorator: _random_mutation_operator: num_test=0 Unsupported set of
arguments: The combination of penalty='l2' and loss='epsilon_insensitive' are
not supported when dual=False, Parameters: penalty='l2',
loss='epsilon_insensitive', dual=False
_pre_test decorator: _random_mutation_operator: num_test=0 Expected n_neighbors
<= n_samples,  but n_samples = 50, n_neighbors = 56
Generation 2 - Current Pareto front scores:
-1      -550.3739539547902      ExtraTreesRegressor(input_matrix,
ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.8500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=13,
ExtraTreesRegressor__n_estimators=100)
-2      -546.4928154699974
ExtraTreesRegressor(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False),
ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.7500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=19,
ExtraTreesRegressor__n_estimators=100)
```

```
_pre_test decorator: _random_mutation_operator: num_test=0 __init__() got an
unexpected keyword argument 'alpha'
_pre_test decorator: _random_mutation_operator: num_test=0 l1 was provided as
affinity. Ward can only work with euclidean distances.
_pre_test decorator: _random_mutation_operator: num_test=0 array must not
contain infs or NaNs
_pre_test decorator: _mate_operator: num_test=0 Expected n_neighbors <=
n_samples,  but n_samples = 50, n_neighbors = 85
_pre_test decorator: _random_mutation_operator: num_test=0 Found array with 0
feature(s) (shape=(50, 0)) while a minimum of 1 is required.
Generation 3 - Current Pareto front scores:
-1      -550.0564698088261      XGBRegressor(input_matrix,
XGBRegressor__learning_rate=0.1, XGBRegressor__max_depth=9,
XGBRegressor__min_child_weight=15, XGBRegressor__n_estimators=100,
XGBRegressor__nthread=1, XGBRegressor__subsample=0.7000000000000001)
-2      -546.4928154699974
ExtraTreesRegressor(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False),
ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.7500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=19,
ExtraTreesRegressor__n_estimators=100)

_pre_test decorator: _random_mutation_operator: num_test=0 __init__() got an
unexpected keyword argument 'alpha'
_pre_test decorator: _random_mutation_operator: num_test=0 X contains negative
values.
_pre_test decorator: _random_mutation_operator: num_test=0 __init__() got an
unexpected keyword argument 'alpha'
Generation 4 - Current Pareto front scores:
-1      -547.752297244712       ExtraTreesRegressor(CombineDFs(input_matrix,
input_matrix), ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.8500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=13,
ExtraTreesRegressor__n_estimators=100)
-2      -545.6280879179124      XGBRegressor(SGDRegressor(input_matrix,
SGDRegressor__eta0=0.7000000000000001, SGDRegressor__learning_rate=optimal,
SGDRegressor__loss=epsilon_insensitive, SGDRegressor__penalty=none,
SGDRegressor__power_t=0.35000000000000003), XGBRegressor__learning_rate=0.1,
XGBRegressor__max_depth=9, XGBRegressor__min_child_weight=15,
XGBRegressor__n_estimators=100, XGBRegressor__nthread=1,
XGBRegressor__subsample=0.7000000000000001)

Generation 5 - Current Pareto front scores:
-1      -546.0123410216604      XGBRegressor(input_matrix,
```

```
XGBRegressor__learning_rate=0.1, XGBRegressor__max_depth=9,
XGBRegressor__min_child_weight=7, XGBRegressor__n_estimators=100,
XGBRegressor__nthread=1, XGBRegressor__subsample=0.7000000000000001)
-2      -545.1459922174483
ExtraTreesRegressor(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False),
ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.9000000000000001,
ExtraTreesRegressor__min_samples_leaf=1,
ExtraTreesRegressor__min_samples_split=16,
ExtraTreesRegressor__n_estimators=100)

_pre_test decorator: _random_mutation_operator: num_test=0 __init__() got an
unexpected keyword argument 'alpha'
_pre_test decorator: _random_mutation_operator: num_test=0 Found array with 0
feature(s) (shape=(50, 0)) while a minimum of 1 is required.
_pre_test decorator: _random_mutation_operator: num_test=0 Found array with 0
feature(s) (shape=(50, 0)) while a minimum of 1 is required.
Generation 6 - Current Pareto front scores:
-1      -546.0123410216604      XGBRegressor(input_matrix,
XGBRegressor__learning_rate=0.1, XGBRegressor__max_depth=9,
XGBRegressor__min_child_weight=7, XGBRegressor__n_estimators=100,
XGBRegressor__nthread=1, XGBRegressor__subsample=0.7000000000000001)
-2      -545.1459922174483
ExtraTreesRegressor(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False),
ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.9000000000000001,
ExtraTreesRegressor__min_samples_leaf=1,
ExtraTreesRegressor__min_samples_split=16,
ExtraTreesRegressor__n_estimators=100)
-3      -538.2405176425407
ExtraTreesRegressor(LassoLarsCV(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False), LassoLarsCV__normalize=False),
ExtraTreesRegressor__bootstrap=True,
ExtraTreesRegressor__max_features=0.7500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=19,
ExtraTreesRegressor__n_estimators=100)

_pre_test decorator: _random_mutation_operator: num_test=0 Expected n_neighbors
<= n_samples,  but n_samples = 50, n_neighbors = 52
_pre_test decorator: _random_mutation_operator: num_test=0 l2 was provided as
affinity. Ward can only work with euclidean distances.
Pipeline encountered that has previously been evaluated during the optimization
```

```
process. Using the score from the previous evaluation.
Generation 7 - Current Pareto front scores:
-1      -546.0123410216604      XGBRegressor(input_matrix,
XGBRegressor__learning_rate=0.1, XGBRegressor__max_depth=9,
XGBRegressor__min_child_weight=7, XGBRegressor__n_estimators=100,
XGBRegressor__nthread=1, XGBRegressor__subsample=0.7000000000000001)
-2      -545.1459922174483
ExtraTreesRegressor(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False),
ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.9000000000000001,
ExtraTreesRegressor__min_samples_leaf=1,
ExtraTreesRegressor__min_samples_split=16,
ExtraTreesRegressor__n_estimators=100)
-3      -538.2405176425407
ExtraTreesRegressor(LassoLarsCV(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False), LassoLarsCV__normalize=False),
ExtraTreesRegressor__bootstrap=True,
ExtraTreesRegressor__max_features=0.7500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=19,
ExtraTreesRegressor__n_estimators=100)

_pre_test decorator: _random_mutation_operator: num_test=0 Expected n_neighbors
<= n_samples,  but n_samples = 50, n_neighbors = 74
_pre_test decorator: _random_mutation_operator: num_test=0 __init__() got an
unexpected keyword argument 'alpha'
_pre_test decorator: _random_mutation_operator: num_test=0 __init__() got an
unexpected keyword argument 'alpha'
_pre_test decorator: _random_mutation_operator: num_test=0 No feature in X meets
the variance threshold 0.20000
_pre_test decorator: _random_mutation_operator: num_test=0 Unsupported set of
arguments: The combination of penalty='l2' and loss='epsilon_insensitive' are
not supported when dual=False, Parameters: penalty='l2',
loss='epsilon_insensitive', dual=False
Pipeline encountered that has previously been evaluated during the optimization
process. Using the score from the previous evaluation.
Generation 8 - Current Pareto front scores:
-1      -546.0123410216604      XGBRegressor(input_matrix,
XGBRegressor__learning_rate=0.1, XGBRegressor__max_depth=9,
XGBRegressor__min_child_weight=7, XGBRegressor__n_estimators=100,
XGBRegressor__nthread=1, XGBRegressor__subsample=0.7000000000000001)
-2      -545.1459922174483
ExtraTreesRegressor(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False),
```

```
ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.9000000000000001,
ExtraTreesRegressor__min_samples_leaf=1,
ExtraTreesRegressor__min_samples_split=16,
ExtraTreesRegressor__n_estimators=100)
-3      -538.2405176425407
ExtraTreesRegressor(LassoLarsCV(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False), LassoLarsCV__normalize=False),
ExtraTreesRegressor__bootstrap=True,
ExtraTreesRegressor__max_features=0.7500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=19,
ExtraTreesRegressor__n_estimators=100)

_pre_test decorator: _random_mutation_operator: num_test=0 Unsupported set of
arguments: The combination of penalty='l2' and loss='epsilon_insensitive' are
not supported when dual=False, Parameters: penalty='l2',
loss='epsilon_insensitive', dual=False
Pipeline encountered that has previously been evaluated during the optimization
process. Using the score from the previous evaluation.
Generation 9 - Current Pareto front scores:
-1      -546.0123410216604      XGBRegressor(input_matrix,
XGBRegressor__learning_rate=0.1, XGBRegressor__max_depth=9,
XGBRegressor__min_child_weight=7, XGBRegressor__n_estimators=100,
XGBRegressor__nthread=1, XGBRegressor__subsample=0.7000000000000001)
-2      -543.5238533407572      XGBRegressor(LinearSVR(input_matrix,
LinearSVR__C=0.001, LinearSVR__dual=False, LinearSVR__epsilon=0.0001,
LinearSVR__loss=squared_epsilon_insensitive, LinearSVR__tol=0.01),
XGBRegressor__learning_rate=0.1, XGBRegressor__max_depth=9,
XGBRegressor__min_child_weight=7, XGBRegressor__n_estimators=100,
XGBRegressor__nthread=1, XGBRegressor__subsample=0.7000000000000001)
-3      -538.2405176425407
ExtraTreesRegressor(LassoLarsCV(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False), LassoLarsCV__normalize=False),
ExtraTreesRegressor__bootstrap=True,
ExtraTreesRegressor__max_features=0.7500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=19,
ExtraTreesRegressor__n_estimators=100)

_pre_test decorator: _random_mutation_operator: num_test=0 Found array with 0
feature(s) (shape=(50, 0)) while a minimum of 1 is required.
_pre_test decorator: _random_mutation_operator: num_test=0 b'[11:14:54]
C:\\Users\\Administrator\\Desktop\\xgboost\\src\\learner.cc:722: Check failed:
mparam_.num_feature != 0 (0 vs. 0) 0 feature is supplied.  Are you using raw
Booster interface?'
```

```
_pre_test decorator: _random_mutation_operator: num_test=0 X contains negative
values.
Invalid pipeline encountered. Skipping its evaluation.
Generation 10 - Current Pareto front scores:
-1      -546.0123410216604      XGBRegressor(input_matrix,
XGBRegressor__learning_rate=0.1, XGBRegressor__max_depth=9,
XGBRegressor__min_child_weight=7, XGBRegressor__n_estimators=100,
XGBRegressor__nthread=1, XGBRegressor__subsample=0.7000000000000001)
-2      -542.5793327890012
ExtraTreesRegressor(LinearSVR(CombineDFs(CombineDFs(input_matrix, input_matrix),
input_matrix), LinearSVR__C=25.0, LinearSVR__dual=True, LinearSVR__epsilon=0.01,
LinearSVR__loss=squared_epsilon_insensitive, LinearSVR__tol=0.0001),
ExtraTreesRegressor__bootstrap=False,
ExtraTreesRegressor__max_features=0.8500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=13,
ExtraTreesRegressor__n_estimators=100)
-3      -538.2405176425407
ExtraTreesRegressor(LassoLarsCV(PolynomialFeatures(input_matrix,
PolynomialFeatures__degree=2, PolynomialFeatures__include_bias=False,
PolynomialFeatures__interaction_only=False), LassoLarsCV__normalize=False),
ExtraTreesRegressor__bootstrap=True,
ExtraTreesRegressor__max_features=0.7500000000000001,
ExtraTreesRegressor__min_samples_leaf=3,
ExtraTreesRegressor__min_samples_split=19,
ExtraTreesRegressor__n_estimators=100)
```

[54]:
```
TPOTRegressor(config_dict={'sklearn.ensemble.AdaBoostRegressor':
{'n_estimators': [100], 'learning_rate': [0.001, 0.01, 0.1, 0.5, 1.0], 'loss':
['linear', 'square', 'exponential']}, 'sklearn.ensemble.ExtraTreesRegressor':
{'n_estimators': [100], 'max_features': array([0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3
, 0.35, 0.4...3 , 0.35, 0.4 , 0.45, 0.5 , 0.55,
       0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95, 1.  ])}}}},
       crossover_rate=0.1, cv=5, disable_update_check=False,
       early_stop=None, generations=10, max_eval_time_mins=5,
       max_time_mins=None, memory=None, mutation_rate=0.9, n_jobs=1,
       offspring_size=None, periodic_checkpoint_folder=None,
       population_size=50, random_state=None,
       scoring=make_scorer(negative_rmse), subsample=1.0, use_dask=True,
       verbosity=3, warm_start=False)
```

[56]:
```
tpot.score(test_features, test_target)
```

[56]:
```
-511.54164310220466
```

[ ]: