Amey

December 13, 2021

0.1 Approach

- load Pandas DataFrame containing electricity data
- split the data in train, test and validation sets (+ normalise independent variables if required)
- fit model parameters using GridSearchCV scikit-learn
- evaluate estimator performance by means of 5 fold 'shuffled' nested cross-validation
- predict cross validated estimates of y for each data point and plot on scatter diagram vs true y
- find the best model and fit to validation set to find electricity demand

0.2 Packages required

- Python 3.8
- Matplotlib
- Pandas
- Numpy
- scikit-learn

0.3 Implement

Install packages

[5]:
```
!pip install scikit-learn
!pip install xgboost
```

```
Requirement already satisfied: scikit-learn in
/Users/maido/PycharmProjects/pythonProject1/venv/lib/python3.8/site-packages
(1.0.1)
Requirement already satisfied: joblib>=0.11 in
/Users/maido/PycharmProjects/pythonProject1/venv/lib/python3.8/site-packages
(from scikit-learn) (1.1.0)
Requirement already satisfied: scipy>=1.1.0 in
/Users/maido/PycharmProjects/pythonProject1/venv/lib/python3.8/site-packages
(from scikit-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Users/maido/PycharmProjects/pythonProject1/venv/lib/python3.8/site-packages
(from scikit-learn) (3.0.0)
```

```
Requirement already satisfied: numpy>=1.14.6 in
/Users/maido/PycharmProjects/pythonProject1/venv/lib/python3.8/site-packages
(from scikit-learn) (1.21.4)
Requirement already satisfied: xgboost in
/Users/maido/PycharmProjects/pythonProject1/venv/lib/python3.8/site-packages
(1.5.1)
Requirement already satisfied: scipy in
/Users/maido/PycharmProjects/pythonProject1/venv/lib/python3.8/site-packages
(from xgboost) (1.7.3)
Requirement already satisfied: numpy in
/Users/maido/PycharmProjects/pythonProject1/venv/lib/python3.8/site-packages
(from xgboost) (1.21.4)
```

[6]:
```
import warnings
warnings.filterwarnings('ignore')
# warnings.filterwarnings(action='once')
```

[7]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split, GridSearchCV,␣
↪cross_val_score, cross_val_predict, KFold
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.pipeline import Pipeline
import xgboost as xgb
```

Preprocessing

- Read the dataset

[8]:
```
file = pd.read_csv('Data.csv')
file = pd.DataFrame(file)
file
```

[8]:

	period	temperature	hours before sunrise	hours before sunset	demand
0	1	8.4	6.016667	17.633333	496.0
1	2	8.1	5.516667	17.133333	535.0
2	3	7.8	5.016667	16.633333	511.0
3	4	7.5	4.516667	16.133333	496.0
4	5	7.3	4.016667	15.633333	490.0
…	…	…	…	…	…
52555	52556	12.4	-15.516667	-3.800000	NaN

```
52556   52557         12.3           -16.016667           -4.300000     NaN
52557   52558         12.2           -16.516667           -4.800000     NaN
52558   52559         11.9           -17.016667           -5.300000     NaN
52559   52560         11.9           -17.516667           -5.800000     NaN

[52560 rows x 5 columns]
```

- Split to train, test and validation datasets

[9]:
```
df = file[file.demand.notnull()]
df
```

[9]:
```
       period  temperature  hours before sunrise  hours before sunset  demand
0           1          8.4              6.016667            17.633333   496.0
1           2          8.1              5.516667            17.133333   535.0
2           3          7.8              5.016667            16.633333   511.0
3           4          7.5              4.516667            16.133333   496.0
4           5          7.3              4.016667            15.633333   490.0
…       …           …                    …                   …      …
48235   48236         13.2           -17.666667            -1.183333   998.0
48236   48237         12.1           -18.166667            -1.683333   867.0
48237   48238         12.1           -18.666667            -2.183333   730.0
48238   48239         12.1           -19.166667            -2.683333   608.0
48239   48240         12.0           -19.666667            -3.183333   517.0

[48240 rows x 5 columns]
```

[10]:
```
y = df.demand
y
```

[10]:
```
0        496.0
1        535.0
2        511.0
3        496.0
4        490.0
         …
48235    998.0
48236    867.0
48237    730.0
48238    608.0
48239    517.0
Name: demand, Length: 48240, dtype: float64
```

- Drop period and demand

[11]:
```
X = df.drop('period', axis=1).drop('demand', axis = 1)
X
```

[11]:

	temperature	hours before sunrise	hours before sunset
0	8.4	6.016667	17.633333
1	8.1	5.516667	17.133333
2	7.8	5.016667	16.633333
3	7.5	4.516667	16.133333
4	7.3	4.016667	15.633333
…	…	…	…
48235	13.2	-17.666667	-1.183333
48236	12.1	-18.166667	-1.683333
48237	12.1	-18.666667	-2.183333
48238	12.1	-19.166667	-2.683333
48239	12.0	-19.666667	-3.183333

[48240 rows x 3 columns]

- Train/test split

[12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
↪random_state=0)
```

- Validation set

[13]:
```
df_vali = file[file.demand.isnull()]
```

[14]:
```
vali = df_vali.drop('period', axis=1).drop('demand', axis = 1)
vali
```

[14]:

	temperature	hours before sunrise	hours before sunset
48240	11.9	3.833333	20.316667
48241	12.0	3.333333	19.816667
48242	12.1	2.833333	19.316667
48243	12.0	2.333333	18.816667
48244	11.9	1.833333	18.316667
…	…	…	…
52555	12.4	-15.516667	-3.800000
52556	12.3	-16.016667	-4.300000
52557	12.2	-16.516667	-4.800000
52558	11.9	-17.016667	-5.300000
52559	11.9	-17.516667	-5.800000

[4320 rows x 3 columns]

Defind the pipeline models

- defind pipeline
- cross validation
- show model coefficients or feature importances
- plot predicted demand vs actual demand

- fit the validation set

[20]:
```
def model(pipeline, parameters, X_train, y_train, X, y):

    grid_obj = GridSearchCV(estimator=pipeline,
                            param_grid=parameters,
                            cv=3,
                            scoring='r2',
                            verbose=2,
                            n_jobs=1,
                            refit=True)
    grid_obj.fit(X_train, y_train)

    grid_obj.predict(vali)


    '''Results'''

    results = pd.DataFrame(pd.DataFrame(grid_obj.cv_results_))
#     results_sorted = results.sort_values(by=['mean_test_score'],␣
↪ascending=False)
    results_vali = grid_obj.predict(vali)


    print("##### Results")
#     print(results_sorted)
    print(results)
    print(results_vali)

    print("best_index", grid_obj.best_index_)
    print("best_score", grid_obj.best_score_)
    print("best_params", grid_obj.best_params_)


    '''Cross Validation'''
    # Cross-validation is a resampling procedure used to evaluate machine␣
↪learning models on a limited data sample.

    estimator = grid_obj.best_estimator_
    '''
    if estimator.named_steps['scl'] == True:
        X = (X - X.mean()) / (X.std())
        y = (y - y.mean()) / (y.std())
    '''
    shuffle = KFold(n_splits=5,
                    shuffle=True,
                    random_state=0)
```

```
    cv_scores = cross_val_score(estimator,
                                X,
                                y.values.ravel(),
                                cv=shuffle,
                                scoring='r2')
    print("##### CV Results")
    print("mean_score", cv_scores.mean())


    '''Show model coefficients or feature importances'''
    # Feature importance refers to how useful a feature is at predicting a␣
↪target variable.
    # A coefficient refers to a number or quantity placed with a variable.

    try:
        print("Model coefficients: ", list(zip(list(X), estimator.
↪named_steps['clf'].coef_)))
    except:
        print("Model does not support model coefficients")

    try:
        print("Feature importances: ", list(zip(list(X), estimator.
↪named_steps['clf'].feature_importances_)))
    except:
        print("Model does not support feature importances")


    '''Predict y vs y_predicted in scatter'''

    y_pred = cross_val_predict(estimator, X, y, cv=shuffle)

    plt.scatter(y, y_pred)
    xmin, xmax = plt.xlim()
    ymin, ymax = plt.ylim()
    plt.plot([xmin, xmax], [ymin, ymax], "g--", lw=1, alpha=0.4)
    plt.xlabel("True demand")
    plt.ylabel("Predicted demand")
    plt.annotate(' R-squared CV = {}'.format(round(float(cv_scores.mean()),␣
↪3)), size=9,
                xy=(xmin,ymax), xytext=(10, -15), textcoords='offset points')
    plt.annotate(grid_obj.best_params_, size=9,
                   xy=(xmin, ymax), xytext=(10, -35), textcoords='offset points',␣
↪wrap=True)
    plt.title('predicted demand vs actual demand')
    plt.show()
```

```
    '''Fit the validation set'''

    # convert array to serial
    vali_series = pd.Series(results_vali)

    df_vali.iloc[:,4] = vali_series.values
    print(df_vali)
```

Pipeline and Parameters

- Linear Regression

[21]:
```
pipe_ols = Pipeline([('scl', StandardScaler()),
                     ('clf', LinearRegression())])

param_ols = {}
```

- XGBoost

[22]:
```
# - XGBoost

pipe_xgb = Pipeline([('scl', StandardScaler()),
                     ('clf', xgb.XGBRegressor())])

param_xgb = {'clf__max_depth':[5],
             'clf__min_child_weight':[6],
             'clf__gamma':[0.01],
             'clf__subsample':[0.7],
             'clf__colsample_bytree':[1]}
```

- KNN

[23]:
```
pipe_knn = Pipeline([('scl', StandardScaler()),
                     ('clf', KNeighborsRegressor())])

param_knn = {'clf__n_neighbors':[5, 10, 15, 25, 30]}
```

- Lasso

[24]:
```
pipe_lasso = Pipeline([('scl', StandardScaler()),
                       ('clf', Lasso(max_iter=1500))])

param_lasso = {'clf__alpha': [0.01, 0.1, 1, 10]}
```

- Ridge

[25]:
```
pipe_ridge = Pipeline([('scl', StandardScaler()),
                       ('clf', Ridge())])
```

```
param_ridge = {'clf__alpha': [0.01, 0.1, 1, 10]}
```

- Polynomial Regression

[26]:
```
pipe_poly = Pipeline([('scl', StandardScaler()),
                      ('polynomial', PolynomialFeatures()),
                      ('clf', LinearRegression())])

param_poly = {'polynomial__degree': [2, 4, 6]}
```

- Decision Tree Regression

[28]:
```
pipe_tree = Pipeline([('scl', StandardScaler()),
                      ('clf', DecisionTreeRegressor())])

param_tree = {'clf__max_depth': [2, 5, 10],
              'clf__min_samples_leaf': [5,10,50,100]}
```

- Random Forest

[29]:
```
pipe_forest = Pipeline([('scl', StandardScaler()),
                        ('clf', RandomForestRegressor())])

param_forest = {'clf__n_estimators': [10, 20, 50],
                'clf__max_features': [None, 1, 2],
                'clf__max_depth': [1, 2, 5]}
```

- MLP Regression

[30]:
```
pipe_neural = Pipeline([('scl', StandardScaler()),
                        ('clf', MLPRegressor())])

param_neural = {'clf__alpha': [0.001, 0.01, 0.1, 1, 10, 100],
                'clf__hidden_layer_sizes': [(5),(10,10),(7,7,7)],
                'clf__solver': ['lbfgs'],
                'clf__activation': ['relu', 'tanh'],
                'clf__learning_rate' : ['constant', 'invscaling']}
```

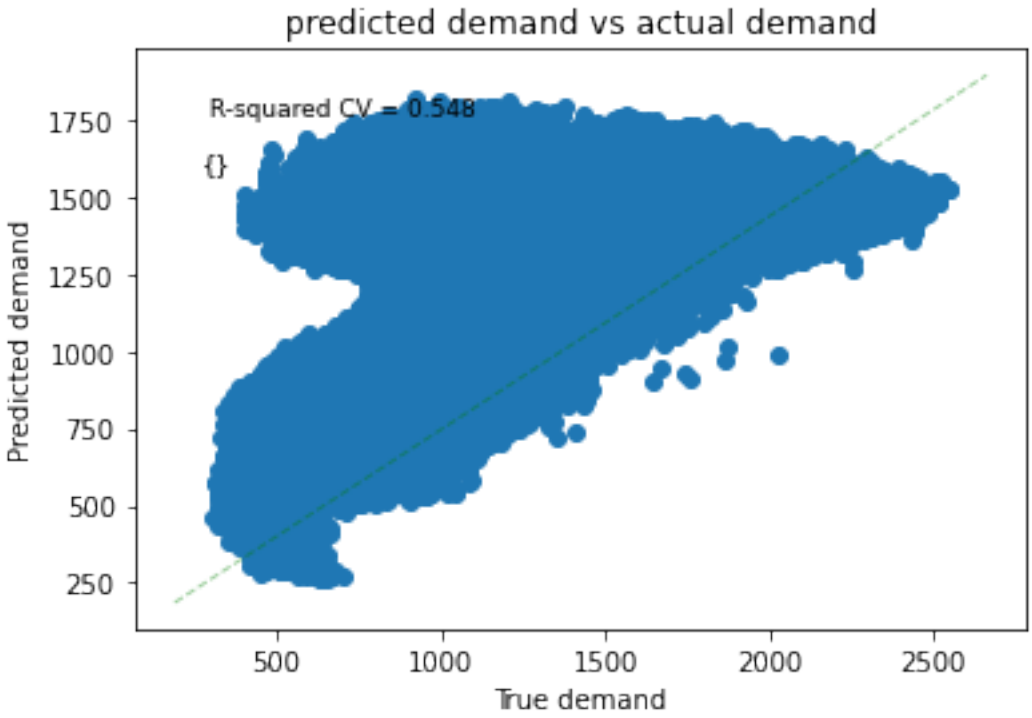
Execute model hyperparameter tuning and crossvalidation

- Linear Regression

[31]:
```
model(pipe_ols, param_ols, X_train, y_train, X, y)
```

```
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV] END … total time=   0.0s
[CV] END … total time=   0.0s
```

```
[CV] END … total time=   0.0s
##### Results
   mean_fit_time  std_fit_time  mean_score_time  std_score_time params  \
0       0.013831      0.002873         0.004555         0.00103     {}

   split0_test_score  split1_test_score  split2_test_score  mean_test_score  \
0           0.547489           0.552356           0.545854         0.548566

   std_test_score  rank_test_score
0        0.002762                1
[ 321.4481177   342.15489827  362.86167883 … 1489.06422134 1513.34921922
 1534.95055411]
best_index 0
best_score 0.548566322512026
best_params {}
##### CV Results
mean_score 0.5478994436082716
Model coefficients:  [('temperature', -55.73826968410802), ('hours before
sunrise', 126.97687725007376), ('hours before sunset', -433.2156285659945)]
Model does not support feature importances
```




```
       period  temperature  hours before sunrise  hours before sunset  \
48240   48241         11.9              3.833333            20.316667
48241   48242         12.0              3.333333            19.816667
```

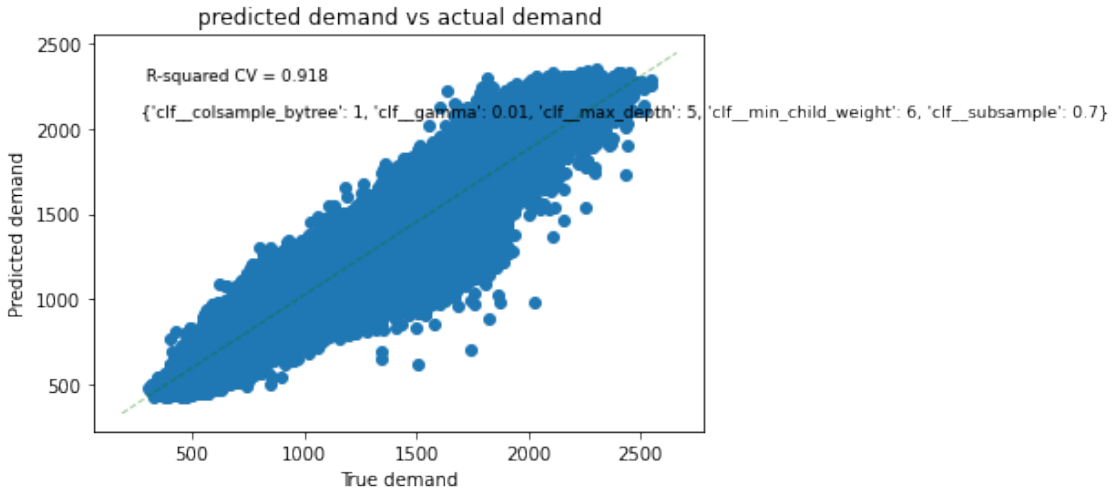
```
48242   48243         12.1              2.833333            19.316667
48243   48244         12.0              2.333333            18.816667
48244   48245         11.9              1.833333            18.316667
…       …          …                    …                   …
52555   52556         12.4            -15.516667            -3.800000
52556   52557         12.3            -16.016667            -4.300000
52557   52558         12.2            -16.516667            -4.800000
52558   52559         11.9            -17.016667            -5.300000
52559   52560         11.9            -17.516667            -5.800000

            demand
48240   321.448118
48241   342.154898
48242   362.861679
48243   385.357568
48244   407.853457
…            …
52555  1444.072443
52556  1466.568332
52557  1489.064221
52558  1513.349219
52559  1534.950554

[4320 rows x 5 columns]
```

- XGBoost

[34]:
```
model(pipe_xgb, param_xgb, X_train, y_train, X, y)
```

```
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV] END clf__colsample_bytree=1, clf__gamma=0.01, clf__max_depth=5,
clf__min_child_weight=6, clf__subsample=0.7; total time=   0.7s
[CV] END clf__colsample_bytree=1, clf__gamma=0.01, clf__max_depth=5,
clf__min_child_weight=6, clf__subsample=0.7; total time=   0.7s
[CV] END clf__colsample_bytree=1, clf__gamma=0.01, clf__max_depth=5,
clf__min_child_weight=6, clf__subsample=0.7; total time=   0.7s
##### Results
   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0        0.67986      0.027707         0.013939        0.000262

  param_clf__colsample_bytree param_clf__gamma param_clf__max_depth  \
0                           1            0.01                    5

  param_clf__min_child_weight param_clf__subsample  \
0                           6                  0.7

                                              params  split0_test_score  \
0  {'clf__colsample_bytree': 1, 'clf__gamma': 0.0…           0.918684
```

```
   split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0           0.914535           0.916465         0.916561        0.001695

   rank_test_score
0                1
[519.5205  514.54376 494.1675  … 825.94946 627.2328  553.7182 ]
best_index 0
best_score 0.9165612487214893
best_params {'clf__colsample_bytree': 1, 'clf__gamma': 0.01, 'clf__max_depth':
5, 'clf__min_child_weight': 6, 'clf__subsample': 0.7}
##### CV Results
mean_score 0.9175378129448346
Model does not support model coefficients
Feature importances:  [('temperature', 0.06672085), ('hours before sunrise',
0.2223035), ('hours before sunset', 0.71097565)]
```




```
       period  temperature  hours before sunrise  hours before sunset  \
48240   48241         11.9              3.833333            20.316667
48241   48242         12.0              3.333333            19.816667
48242   48243         12.1              2.833333            19.316667
48243   48244         12.0              2.333333            18.816667
48244   48245         11.9              1.833333            18.316667
…         …           …                    …                   …
52555   52556         12.4            -15.516667            -3.800000
52556   52557         12.3            -16.016667            -4.300000
52557   52558         12.2            -16.516667            -4.800000
52558   52559         11.9            -17.016667            -5.300000
52559   52560         11.9            -17.516667            -5.800000

            demand
```

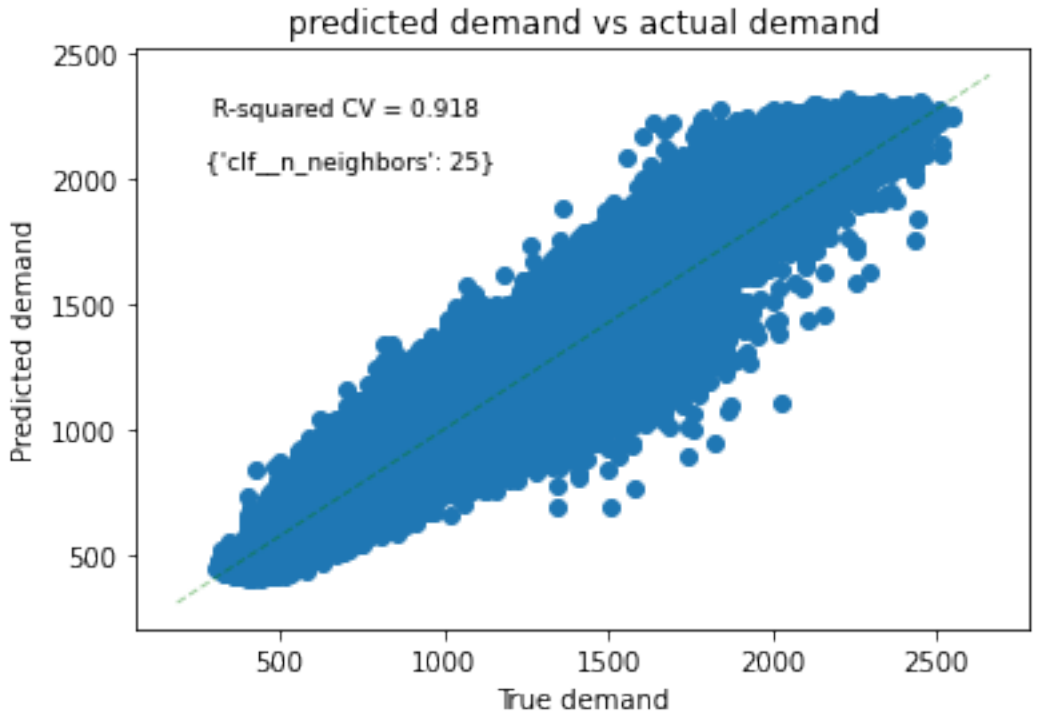
```
48240   519.520508
48241   514.543762
48242   494.167511
48243   485.215851
48244   482.247131
…            …
52555  1154.285400
52556   989.376282
52557   825.949463
52558   627.232788
52559   553.718201

[4320 rows x 5 columns]
```

- KNN

[35]:
```
model(pipe_knn, param_knn, X_train, y_train, X, y)
```

```
Fitting 3 folds for each of 5 candidates, totalling 15 fits
[CV] END …clf__n_neighbors=5; total time=   0.1s
[CV] END …clf__n_neighbors=5; total time=   0.1s
[CV] END …clf__n_neighbors=5; total time=   0.1s
[CV] END …clf__n_neighbors=10; total time=   0.1s
[CV] END …clf__n_neighbors=10; total time=   0.1s
[CV] END …clf__n_neighbors=10; total time=   0.1s
[CV] END …clf__n_neighbors=15; total time=   0.1s
[CV] END …clf__n_neighbors=15; total time=   0.1s
[CV] END …clf__n_neighbors=15; total time=   0.1s
[CV] END …clf__n_neighbors=25; total time=   0.1s
[CV] END …clf__n_neighbors=25; total time=   0.1s
[CV] END …clf__n_neighbors=25; total time=   0.1s
[CV] END …clf__n_neighbors=30; total time=   0.1s
[CV] END …clf__n_neighbors=30; total time=   0.1s
[CV] END …clf__n_neighbors=30; total time=   0.1s
##### Results
   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       0.011664      0.000315         0.043155        0.002428
1       0.011563      0.000460         0.059317        0.002865
2       0.012413      0.000549         0.070815        0.000319
3       0.012232      0.000336         0.091150        0.004666
4       0.016021      0.000279         0.124685        0.003689

  param_clf__n_neighbors                    params  split0_test_score  \
0                      5   {'clf__n_neighbors': 5}           0.908639
1                     10  {'clf__n_neighbors': 10}           0.915317
2                     15  {'clf__n_neighbors': 15}           0.917410
3                     25  {'clf__n_neighbors': 25}           0.917865
4                     30  {'clf__n_neighbors': 30}           0.917344
```

```
   split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0           0.905553           0.907914         0.907369        0.001318
1           0.913781           0.914244         0.914448        0.000643
2           0.915614           0.916473         0.916499        0.000733
3           0.915832           0.916796         0.916831        0.000830
4           0.915362           0.916498         0.916401        0.000812

   rank_test_score
0                5
1                4
2                2
3                1
4                3
[521.12 520.36 486.52 … 852.52 647.44 601.2 ]
best_index 3
best_score 0.9168309762861871
best_params {'clf__n_neighbors': 25}
##### CV Results
mean_score 0.917635589823725
Model does not support model coefficients
Model does not support feature importances
```




```
      period  temperature  hours before sunrise  hours before sunset   demand
```

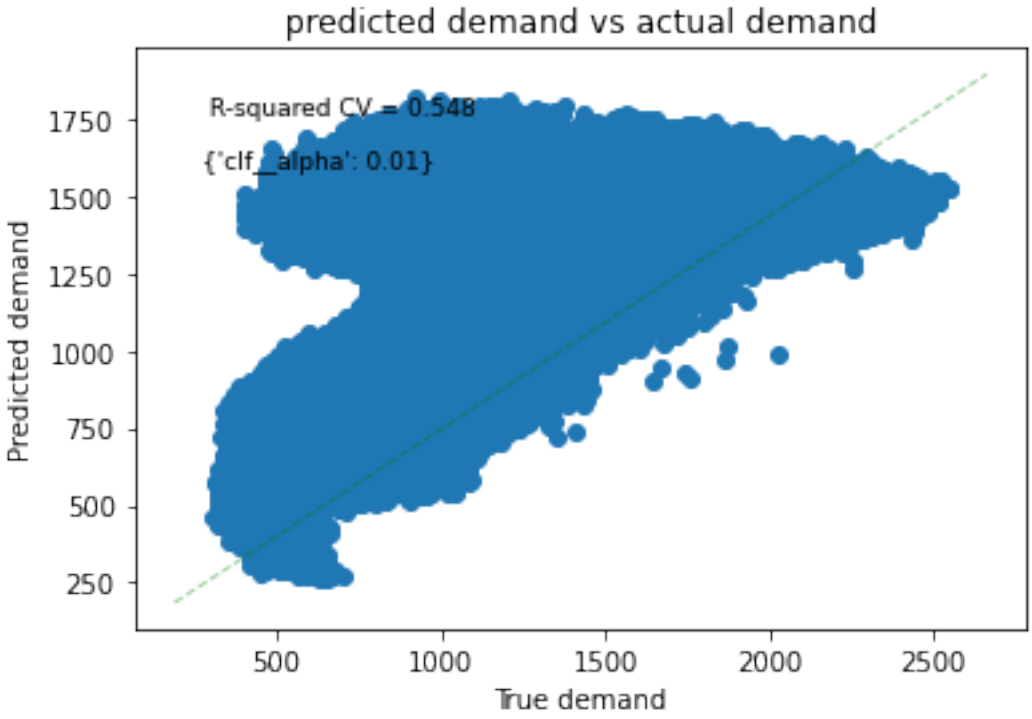
```
48240   48241         11.9              3.833333            20.316667   521.12
48241   48242         12.0              3.333333            19.816667   520.36
48242   48243         12.1              2.833333            19.316667   486.52
48243   48244         12.0              2.333333            18.816667   476.52
48244   48245         11.9              1.833333            18.316667   467.12
…       …          …                    …                  …       …
52555   52556         12.4            -15.516667            -3.800000  1156.68
52556   52557         12.3            -16.016667            -4.300000  1043.80
52557   52558         12.2            -16.516667            -4.800000   852.52
52558   52559         11.9            -17.016667            -5.300000   647.44
52559   52560         11.9            -17.516667            -5.800000   601.20

[4320 rows x 5 columns]
```

- Lasso

[36]:
```
model(pipe_lasso, param_lasso, X_train, y_train, X, y)
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[CV] END …clf__alpha=0.01; total time=   0.0s
[CV] END …clf__alpha=0.01; total time=   0.0s
[CV] END …clf__alpha=0.01; total time=   0.0s
[CV] END …clf__alpha=0.1; total time=   0.0s
[CV] END …clf__alpha=0.1; total time=   0.0s
[CV] END …clf__alpha=0.1; total time=   0.0s
[CV] END …clf__alpha=1; total time=   0.0s
[CV] END …clf__alpha=1; total time=   0.0s
[CV] END …clf__alpha=1; total time=   0.0s
[CV] END …clf__alpha=10; total time=   0.0s
[CV] END …clf__alpha=10; total time=   0.0s
[CV] END …clf__alpha=10; total time=   0.0s
##### Results
   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       0.019372      0.003883         0.002942        0.000292
1       0.011580      0.000126         0.002811        0.000289
2       0.010245      0.000410         0.002864        0.000270
3       0.008257      0.000929         0.002637        0.000314

  param_clf__alpha                params  split0_test_score  \
0             0.01  {'clf__alpha': 0.01}           0.547490
1              0.1   {'clf__alpha': 0.1}           0.547501
2                1     {'clf__alpha': 1}           0.547481
3               10    {'clf__alpha': 10}           0.538635

   split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0           0.552357           0.545851         0.548566        0.002763
1           0.552367           0.545827         0.548565        0.002774
2           0.552332           0.545458         0.548424        0.002885
```

```
3           0.543320           0.535048          0.539001         0.003387

   rank_test_score
0                1
1                2
2                3
3                4
[ 321.52964041  342.23628052  362.94292063 … 1489.05529693 1513.34207834
 1534.94375378]
best_index 0
best_score 0.5485663040582126
best_params {'clf__alpha': 0.01}
##### CV Results
mean_score 0.5478994297681836
Model coefficients:  [('temperature', -55.76823988289015), ('hours before
sunrise', 126.81872246453133), ('hours before sunset', -433.0625589484212)]
Model does not support feature importances
```




```
       period  temperature  hours before sunrise  hours before sunset  \
48240   48241         11.9              3.833333            20.316667
48241   48242         12.0              3.333333            19.816667
48242   48243         12.1              2.833333            19.316667
48243   48244         12.0              2.333333            18.816667
48244   48245         11.9              1.833333            18.316667
```

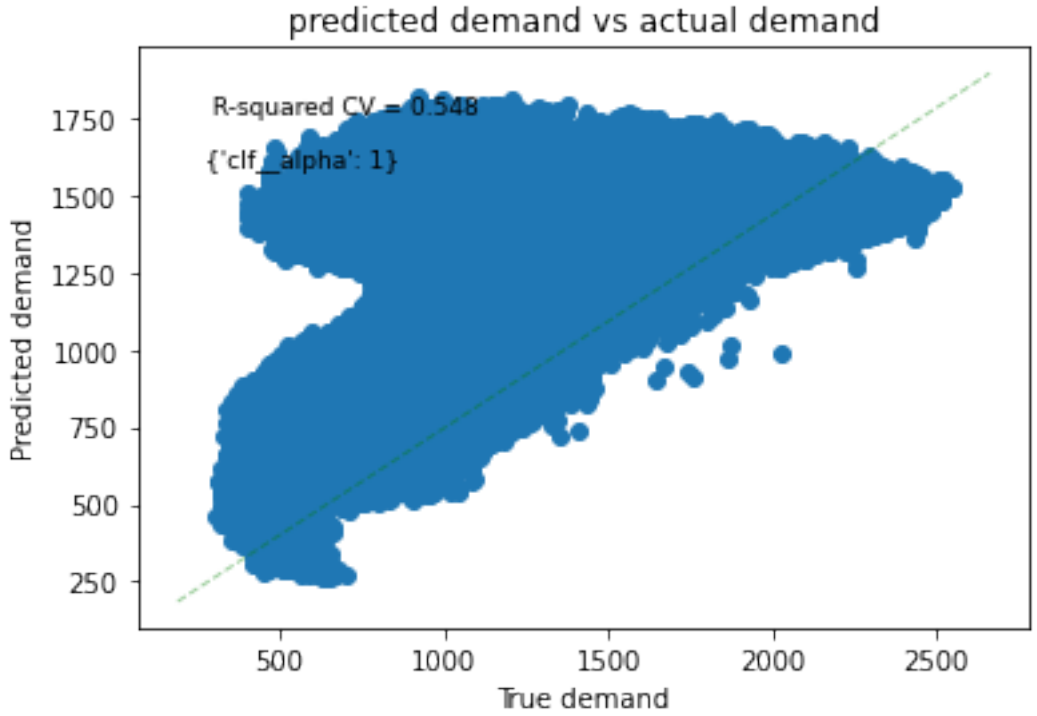
```
…       …           …                    …                    …
52555   52556         12.4            -15.516667            -3.800000
52556   52557         12.3            -16.016667            -4.300000
52557   52558         12.2            -16.516667            -4.800000
52558   52559         11.9            -17.016667            -5.300000
52559   52560         11.9            -17.516667            -5.800000

            demand
48240   321.529640
48241   342.236281
48242   362.942921
48243   385.439631
48244   407.936342
…              …
52555  1444.061875
52556  1466.558586
52557  1489.055297
52558  1513.342078
52559  1534.943754

[4320 rows x 5 columns]
```

- Ridge

[37]:
```
model(pipe_ridge, param_ridge, X_train, y_train, X, y)
```

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
[CV] END …clf__alpha=0.01; total time=   0.0s
[CV] END …clf__alpha=0.01; total time=   0.0s
[CV] END …clf__alpha=0.01; total time=   0.0s
[CV] END …clf__alpha=0.1; total time=   0.0s
[CV] END …clf__alpha=0.1; total time=   0.0s
[CV] END …clf__alpha=0.1; total time=   0.0s
[CV] END …clf__alpha=1; total time=   0.0s
[CV] END …clf__alpha=1; total time=   0.0s
[CV] END …clf__alpha=1; total time=   0.0s
[CV] END …clf__alpha=10; total time=   0.0s
[CV] END …clf__alpha=10; total time=   0.0s
[CV] END …clf__alpha=10; total time=   0.0s
##### Results
   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       0.008222      0.001759         0.002181        0.000303
1       0.007092      0.001393         0.002658        0.000653
2       0.005797      0.000824         0.001775        0.000040
3       0.005965      0.001369         0.002207        0.000375

  param_clf__alpha                params  split0_test_score  \
0             0.01  {'clf__alpha': 0.01}           0.547489
```

```
1              0.1   {'clf__alpha': 0.1}           0.547489
2                1     {'clf__alpha': 1}           0.547491
3               10    {'clf__alpha': 10}           0.547506

   split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0           0.552356           0.545854         0.548566        0.002762
1           0.552356           0.545853         0.548566        0.002762
2           0.552358           0.545850         0.548566        0.002763
3           0.552368           0.545818         0.548564        0.002777

   rank_test_score
0                3
1                2
2                1
3                4
[ 321.53000135  342.2362621   362.94252284 … 1489.04783699 1513.33458552
 1534.93596821]
best_index 2
best_score 0.5485663301961063
best_params {'clf__alpha': 1}
##### CV Results
mean_score 0.547899440883825
Model coefficients:  [('temperature', -55.77363688870504), ('hours before
sunrise', 126.82937131627627), ('hours before sunset', -433.0690376983775)]
Model does not support feature importances
```



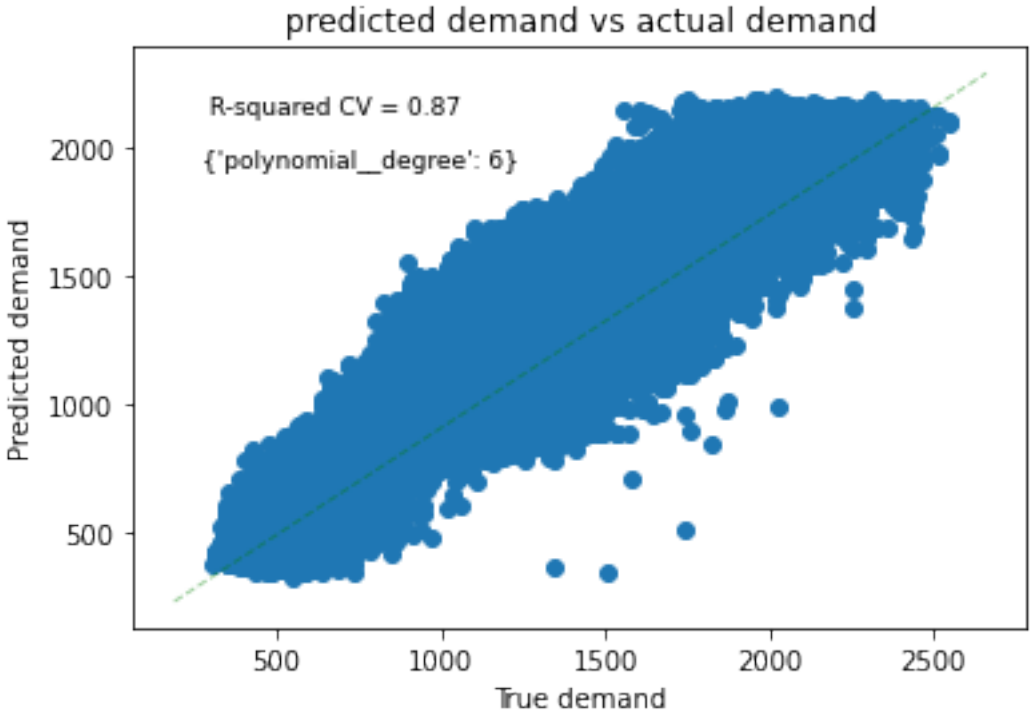
```
       period  temperature  hours before sunrise  hours before sunset  \
48240   48241         11.9              3.833333            20.316667
48241   48242         12.0              3.333333            19.816667
48242   48243         12.1              2.833333            19.316667
48243   48244         12.0              2.333333            18.816667
48244   48245         11.9              1.833333            18.316667
…         …           …                    …                   …
52555   52556         12.4            -15.516667            -3.800000
52556   52557         12.3            -16.016667            -4.300000
52557   52558         12.2            -16.516667            -4.800000
52558   52559         11.9            -17.016667            -5.300000
52559   52560         11.9            -17.516667            -5.800000

            demand
48240   321.530001
48241   342.236262
48242   362.942523
48243   385.439027
48244   407.935532
…              …
52555  1444.054828
52556  1466.551332
52557  1489.047837
52558  1513.334586
52559  1534.935968

[4320 rows x 5 columns]
```

- Polynomial Regression

[38]:
```
model(pipe_poly, param_poly, X_train, y_train, X, y)
```

```
Fitting 3 folds for each of 3 candidates, totalling 9 fits
[CV] END …polynomial__degree=2; total time=   0.0s
[CV] END …polynomial__degree=2; total time=   0.0s
[CV] END …polynomial__degree=2; total time=   0.0s
[CV] END …polynomial__degree=4; total time=   0.0s
[CV] END …polynomial__degree=4; total time=   0.1s
[CV] END …polynomial__degree=4; total time=   0.1s
[CV] END …polynomial__degree=6; total time=   0.1s
[CV] END …polynomial__degree=6; total time=   0.1s
[CV] END …polynomial__degree=6; total time=   0.1s
##### Results
   mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0       0.017192      0.002551         0.003838        0.000559
1       0.042403      0.001487         0.009976        0.002202
2       0.119862      0.009164         0.010884        0.000906
```

```
  param_polynomial__degree                   params  split0_test_score  \
0                        2  {'polynomial__degree': 2}           0.642505
1                        4  {'polynomial__degree': 4}           0.809408
2                        6  {'polynomial__degree': 6}           0.871406

   split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0           0.649147           0.644655         0.645436        0.002767
1           0.806470           0.810984         0.808954        0.001870
2           0.868596           0.869231         0.869744        0.001203

   rank_test_score
0                3
1                2
2                1
[652.1328581  516.0687481  436.67898674 … 922.87818015 667.18504386
 356.47292377]
best_index 2
best_score 0.8697440760087667
best_params {'polynomial__degree': 6}
##### CV Results
mean_score 0.8695894465627841
Model coefficients:  [('temperature', 8.495889079381798e-08), ('hours before
sunrise', -197.60258201034532), ('hours before sunset', -3.7160343507902347)]
Model does not support feature importances
```



```
       period  temperature  hours before sunrise  hours before sunset  \
48240   48241         11.9              3.833333            20.316667
48241   48242         12.0              3.333333            19.816667
48242   48243         12.1              2.833333            19.316667
48243   48244         12.0              2.333333            18.816667
48244   48245         11.9              1.833333            18.316667
…         …           …                    …                   …
52555   52556         12.4            -15.516667            -3.800000
52556   52557         12.3            -16.016667            -4.300000
52557   52558         12.2            -16.516667            -4.800000
52558   52559         11.9            -17.016667            -5.300000
52559   52560         11.9            -17.516667            -5.800000

            demand
48240   652.132858
48241   516.068748
48242   436.678987
48243   399.641621
48244   395.362234
…              …
52555  1293.980206
52556  1131.264377
52557   922.878180
52558   667.185044
52559   356.472924

[4320 rows x 5 columns]
```

- Decision Tree Regression

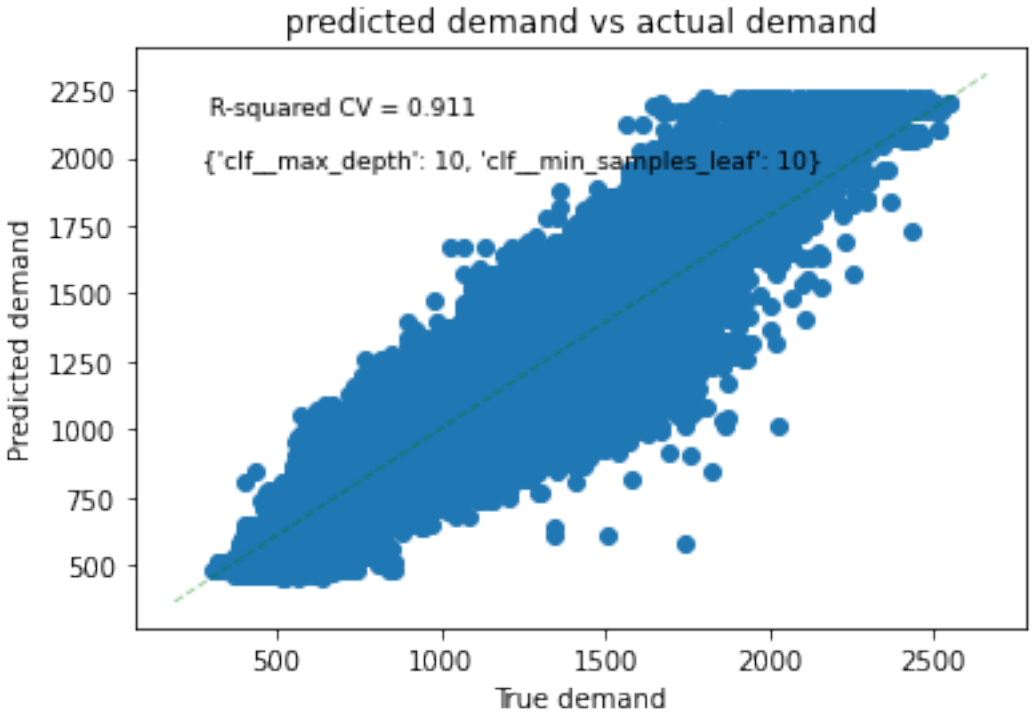
[39]:
```
model(pipe_tree, param_tree, X_train, y_train, X, y)
```

```
Fitting 3 folds for each of 12 candidates, totalling 36 fits
[CV] END …clf__max_depth=2, clf__min_samples_leaf=5; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=5; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=5; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=10; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=10; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=10; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=50; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=50; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=50; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=100; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=100; total time=   0.0s
[CV] END …clf__max_depth=2, clf__min_samples_leaf=100; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=5; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=5; total time=   0.0s
```

```
[CV] END …clf__max_depth=5, clf__min_samples_leaf=5; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=10; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=10; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=10; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=50; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=50; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=50; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=100; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=100; total time=   0.0s
[CV] END …clf__max_depth=5, clf__min_samples_leaf=100; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=5; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=5; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=5; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=10; total time=   0.1s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=10; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=10; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=50; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=50; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=50; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=100; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=100; total time=   0.0s
[CV] END …clf__max_depth=10, clf__min_samples_leaf=100; total time=   0.0s
##### Results
    mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0        0.013890      0.001105         0.003149        0.000674
1        0.012713      0.001272         0.002156        0.000156
2        0.012739      0.000090         0.002294        0.000267
3        0.012082      0.000279         0.002118        0.000075
4        0.023791      0.000964         0.002550        0.000037
5        0.022014      0.000419         0.002433        0.000019
6        0.022409      0.000622         0.002500        0.000079
7        0.023173      0.000880         0.002782        0.000389
8        0.039626      0.000470         0.004073        0.000793
9        0.042504      0.003224         0.003605        0.000175
10       0.039722      0.001088         0.003242        0.000269
11       0.034558      0.000243         0.003091        0.000170

   param_clf__max_depth param_clf__min_samples_leaf  \
0                     2                           5
1                     2                          10
2                     2                          50
3                     2                         100
4                     5                           5
5                     5                          10
6                     5                          50
7                     5                         100
8                    10                           5
9                    10                          10
```

```
10                   10                           50
11                   10                          100

                                                params  split0_test_score  \
0   {'clf__max_depth': 2, 'clf__min_samples_leaf': 5}           0.614903
1   {'clf__max_depth': 2, 'clf__min_samples_leaf':…           0.614903
2   {'clf__max_depth': 2, 'clf__min_samples_leaf':…           0.614903
3   {'clf__max_depth': 2, 'clf__min_samples_leaf':…           0.614903
4   {'clf__max_depth': 5, 'clf__min_samples_leaf': 5}           0.865191
5   {'clf__max_depth': 5, 'clf__min_samples_leaf':…           0.865191
6   {'clf__max_depth': 5, 'clf__min_samples_leaf':…           0.865008
7   {'clf__max_depth': 5, 'clf__min_samples_leaf':…           0.864936
8   {'clf__max_depth': 10, 'clf__min_samples_leaf'…           0.908071
9   {'clf__max_depth': 10, 'clf__min_samples_leaf'…           0.909314
10  {'clf__max_depth': 10, 'clf__min_samples_leaf'…           0.906340
11  {'clf__max_depth': 10, 'clf__min_samples_leaf'…           0.897660

    split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0            0.597650           0.599382         0.603978        0.007757
1            0.597650           0.599382         0.603978        0.007757
2            0.597650           0.599382         0.603978        0.007757
3            0.597650           0.599382         0.603978        0.007757
4            0.862205           0.862235         0.863210        0.001401
5            0.862205           0.862235         0.863210        0.001401
6            0.861967           0.862067         0.863014        0.001411
7            0.861967           0.861958         0.862954        0.001402
8            0.906448           0.907039         0.907186        0.000671
9            0.907410           0.907188         0.907971        0.000954
10           0.905126           0.904799         0.905422        0.000663
11           0.896922           0.897470         0.897351        0.000313

    rank_test_score
0                 9
1                 9
2                 9
3                 9
4                 5
5                 5
6                 7
7                 8
8                 2
9                 1
10                3
11                4
[ 519.18090452  519.18090452  482.85922684 … 1016.53846154  706.15789474
  588.6       ]
best_index 9
best_score 0.9079707582938504
```

```
best_params {'clf__max_depth': 10, 'clf__min_samples_leaf': 10}
##### CV Results
mean_score 0.9110973759686589
Model does not support model coefficients
Feature importances:  [('temperature', 0.049639106318395375), ('hours before
sunrise', 0.22542062166354684), ('hours before sunset', 0.7249402720180578)]
```



	period	temperature	hours before sunrise	hours before sunset	\
48240	48241	11.9	3.833333	20.316667	
48241	48242	12.0	3.333333	19.816667	
48242	48243	12.1	2.833333	19.316667	
48243	48244	12.0	2.333333	18.816667	
48244	48245	11.9	1.833333	18.316667	
…	…	…	…	…	
52555	52556	12.4	-15.516667	-3.800000	
52556	52557	12.3	-16.016667	-4.300000	
52557	52558	12.2	-16.516667	-4.800000	
52558	52559	11.9	-17.016667	-5.300000	
52559	52560	11.9	-17.516667	-5.800000	

	demand
48240	519.180905
48241	519.180905
48242	482.859227

```
48243   482.859227
48244   482.859227
…            …
52555  1211.777778
52556   946.500000
52557  1016.538462
52558   706.157895
52559   588.600000

[4320 rows x 5 columns]
```

- Random Forest

[40]:
```
model(pipe_forest, param_forest, X_train, y_train, X, y)
```

```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=50; total
time=   0.2s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=50; total
time=   0.2s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=50; total
time=   0.2s
[CV] END clf__max_depth=1, clf__max_features=1, clf__n_estimators=10; total
time=   0.0s
[CV] END clf__max_depth=1, clf__max_features=1, clf__n_estimators=10; total
time=   0.0s
[CV] END clf__max_depth=1, clf__max_features=1, clf__n_estimators=10; total
time=   0.0s
[CV] END clf__max_depth=1, clf__max_features=1, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=1, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=1, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=1, clf__n_estimators=50; total
time=   0.2s
```

```
[CV] END clf__max_depth=1, clf__max_features=1, clf__n_estimators=50; total
time=   0.2s
[CV] END clf__max_depth=1, clf__max_features=1, clf__n_estimators=50; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=2, clf__n_estimators=10; total
time=   0.0s
[CV] END clf__max_depth=1, clf__max_features=2, clf__n_estimators=10; total
time=   0.0s
[CV] END clf__max_depth=1, clf__max_features=2, clf__n_estimators=10; total
time=   0.0s
[CV] END clf__max_depth=1, clf__max_features=2, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=2, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=2, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=1, clf__max_features=2, clf__n_estimators=50; total
time=   0.2s
[CV] END clf__max_depth=1, clf__max_features=2, clf__n_estimators=50; total
time=   0.2s
[CV] END clf__max_depth=1, clf__max_features=2, clf__n_estimators=50; total
time=   0.2s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=20; total
time=   0.2s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=20; total
time=   0.2s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=20; total
time=   0.2s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=50; total
time=   0.4s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=50; total
time=   0.4s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=50; total
time=   0.4s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=20; total
time=   0.1s
```

```
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=50; total
time=   0.3s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=50; total
time=   0.3s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=50; total
time=   0.3s
[CV] END clf__max_depth=2, clf__max_features=2, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=2, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=2, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=2, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=2, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=2, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=2, clf__n_estimators=50; total
time=   0.3s
[CV] END clf__max_depth=2, clf__max_features=2, clf__n_estimators=50; total
time=   0.3s
[CV] END clf__max_depth=2, clf__max_features=2, clf__n_estimators=50; total
time=   0.3s
[CV] END clf__max_depth=5, clf__max_features=None, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=None, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=None, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=None, clf__n_estimators=20; total
time=   0.3s
[CV] END clf__max_depth=5, clf__max_features=None, clf__n_estimators=20; total
time=   0.3s
[CV] END clf__max_depth=5, clf__max_features=None, clf__n_estimators=20; total
time=   0.3s
[CV] END clf__max_depth=5, clf__max_features=None, clf__n_estimators=50; total
time=   0.7s
[CV] END clf__max_depth=5, clf__max_features=None, clf__n_estimators=50; total
time=   0.7s
[CV] END clf__max_depth=5, clf__max_features=None, clf__n_estimators=50; total
time=   0.7s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=10; total
time=   0.1s
```

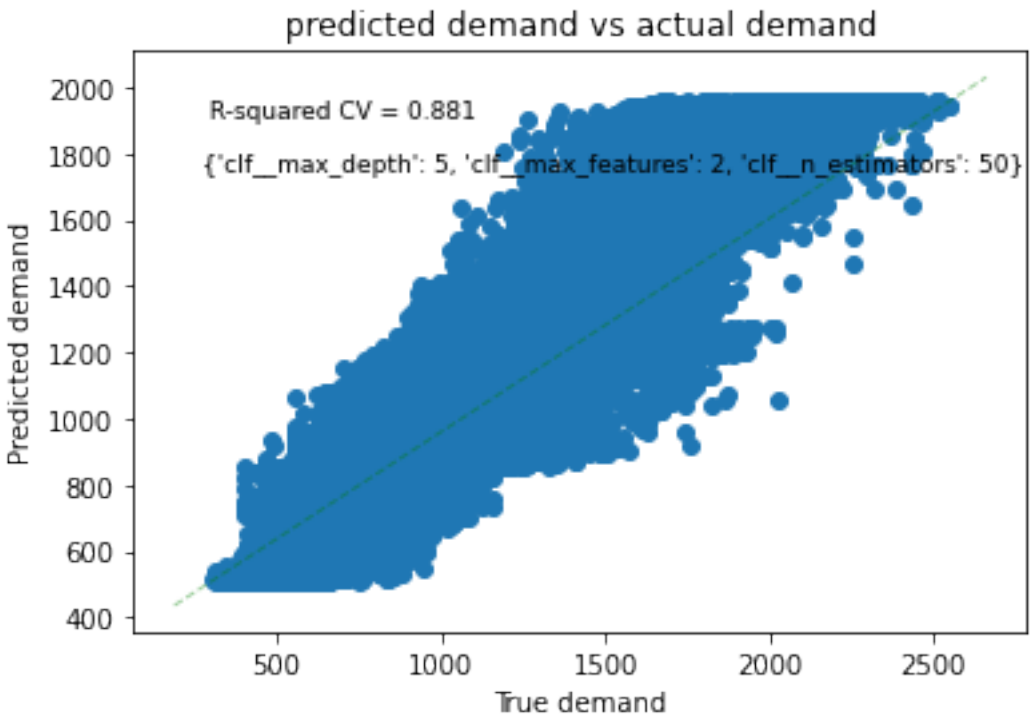
```
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=20; total
time=   0.2s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=20; total
time=   0.2s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=50; total
time=   0.4s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=50; total
time=   0.4s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=50; total
time=   0.4s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=10; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=20; total
time=   0.2s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=20; total
time=   0.2s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=20; total
time=   0.2s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=50; total
time=   0.5s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=50; total
time=   0.5s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=50; total
time=   0.5s
##### Results
    mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0        0.050440      0.002979         0.004551        0.000111
1        0.090407      0.002945         0.007325        0.000499
2        0.208755      0.002772         0.014575        0.000845
3        0.032132      0.001067         0.004631        0.000299
4        0.059788      0.003968         0.007571        0.000275
5        0.140067      0.006233         0.013916        0.001096
6        0.038643      0.000866         0.004786        0.000152
7        0.079595      0.003867         0.008774        0.000235
8        0.195599      0.016746         0.014194        0.001196
9        0.099065      0.005290         0.007995        0.000705
10       0.167137      0.010101         0.010642        0.002517
11       0.369902      0.008323         0.020400        0.001353
```

```
12       0.047767      0.002272         0.005835        0.000270
13       0.092982      0.004028         0.009894        0.000881
14       0.233952      0.001922         0.022782        0.002032
15       0.061241      0.006197         0.006045        0.000703
16       0.109946      0.001914         0.009367        0.001197
17       0.264524      0.005760         0.017078        0.000335
18       0.140899      0.001377         0.007356        0.000197
19       0.280432      0.002718         0.012015        0.000318
20       0.696938      0.013742         0.026987        0.001679
21       0.075595      0.001211         0.008137        0.001339
22       0.139207      0.002088         0.011742        0.000274
23       0.367554      0.028901         0.028096        0.003817
24       0.109818      0.005322         0.007892        0.001335
25       0.206547      0.005175         0.011856        0.000236
26       0.509565      0.003708         0.025033        0.000330

   param_clf__max_depth param_clf__max_features param_clf__n_estimators  \
0                     1                    None                      10
1                     1                    None                      20
2                     1                    None                      50
3                     1                       1                      10
4                     1                       1                      20
5                     1                       1                      50
6                     1                       2                      10
7                     1                       2                      20
8                     1                       2                      50
9                     2                    None                      10
10                    2                    None                      20
11                    2                    None                      50
12                    2                       1                      10
13                    2                       1                      20
14                    2                       1                      50
15                    2                       2                      10
16                    2                       2                      20
17                    2                       2                      50
18                    5                    None                      10
19                    5                    None                      20
20                    5                    None                      50
21                    5                       1                      10
22                    5                       1                      20
23                    5                       1                      50
24                    5                       2                      10
25                    5                       2                      20
26                    5                       2                      50

                                               params  split0_test_score  \
0   {'clf__max_depth': 1, 'clf__max_features': Non…           0.413660
1   {'clf__max_depth': 1, 'clf__max_features': Non…           0.419199
```

2	{'clf__max_depth': 1, 'clf__max_features': Non…	0.412466
3	{'clf__max_depth': 1, 'clf__max_features': 1, …	0.402958
4	{'clf__max_depth': 1, 'clf__max_features': 1, …	0.355479
5	{'clf__max_depth': 1, 'clf__max_features': 1, …	0.400604
6	{'clf__max_depth': 1, 'clf__max_features': 2, …	0.451589
7	{'clf__max_depth': 1, 'clf__max_features': 2, …	0.462691
8	{'clf__max_depth': 1, 'clf__max_features': 2, …	0.461774
9	{'clf__max_depth': 2, 'clf__max_features': Non…	0.627858
10	{'clf__max_depth': 2, 'clf__max_features': Non…	0.623778
11	{'clf__max_depth': 2, 'clf__max_features': Non…	0.625599
12	{'clf__max_depth': 2, 'clf__max_features': 1, …	0.579902
13	{'clf__max_depth': 2, 'clf__max_features': 1, …	0.537393
14	{'clf__max_depth': 2, 'clf__max_features': 1, …	0.595072
15	{'clf__max_depth': 2, 'clf__max_features': 2, …	0.649296
16	{'clf__max_depth': 2, 'clf__max_features': 2, …	0.665726
17	{'clf__max_depth': 2, 'clf__max_features': 2, …	0.663151
18	{'clf__max_depth': 5, 'clf__max_features': Non…	0.877035
19	{'clf__max_depth': 5, 'clf__max_features': Non…	0.878047
20	{'clf__max_depth': 5, 'clf__max_features': Non…	0.878602
21	{'clf__max_depth': 5, 'clf__max_features': 1, …	0.868269
22	{'clf__max_depth': 5, 'clf__max_features': 1, …	0.833036
23	{'clf__max_depth': 5, 'clf__max_features': 1, …	0.815993
24	{'clf__max_depth': 5, 'clf__max_features': 2, …	0.881611
25	{'clf__max_depth': 5, 'clf__max_features': 2, …	0.882281
26	{'clf__max_depth': 5, 'clf__max_features': 2, …	0.885927

	split1_test_score	split2_test_score	mean_test_score	std_test_score	\
0	0.411217	0.417171	0.414016	0.002444	
1	0.409677	0.417514	0.415463	0.004149	
2	0.411975	0.419371	0.414604	0.003377	
3	0.430615	0.432298	0.421957	0.013452	
4	0.374685	0.355440	0.361868	0.009063	
5	0.379399	0.384187	0.388063	0.009081	
6	0.432193	0.472560	0.452114	0.016484	
7	0.465115	0.460461	0.462756	0.001901	
8	0.458901	0.469348	0.463341	0.004407	
9	0.619052	0.637672	0.628194	0.007605	
10	0.608579	0.631079	0.621145	0.009372	
11	0.618042	0.627947	0.623863	0.004226	
12	0.547181	0.626127	0.584403	0.032386	
13	0.576217	0.593439	0.569016	0.023440	
14	0.615503	0.606490	0.605688	0.008360	
15	0.653027	0.661963	0.654762	0.005315	
16	0.659070	0.669605	0.664801	0.004350	
17	0.656202	0.668181	0.662511	0.004911	
18	0.874166	0.875834	0.875678	0.001176	
19	0.875298	0.878891	0.877412	0.001534	
20	0.875322	0.878904	0.877609	0.001622	

```
21           0.830107          0.825667         0.841348        0.019122
22           0.808146          0.833757         0.824980        0.011907
23           0.827794          0.833571         0.825786        0.007315
24           0.878888          0.880208         0.880236        0.001112
25           0.879210          0.883731         0.881741        0.001885
26           0.879435          0.884950         0.883437        0.002858

    rank_test_score
0                25
1                23
2                24
3                22
4                27
5                26
6                21
7                20
8                19
9                13
10               15
11               14
12               17
13               18
14               16
15               12
16               10
17               11
18                6
19                5
20                4
21                7
22                9
23                8
24                3
25                2
26                1
[ 509.51878878  509.51878878  509.51878878 … 1013.42267422  862.1179964
  776.29349706]
best_index 26
best_score 0.883437242773351
best_params {'clf__max_depth': 5, 'clf__max_features': 2, 'clf__n_estimators':
50}
##### CV Results
mean_score 0.8812342581497843
Model does not support model coefficients
Feature importances:  [('temperature', 0.0719854088877638), ('hours before
sunrise', 0.3693710252629837), ('hours before sunset', 0.5586435658492525)]
```



	period	temperature	hours before sunrise	hours before sunset	\
48240	48241	11.9	3.833333	20.316667	
48241	48242	12.0	3.333333	19.816667	
48242	48243	12.1	2.833333	19.316667	
48243	48244	12.0	2.333333	18.816667	
48244	48245	11.9	1.833333	18.316667	
…	…	…	…	…	
52555	52556	12.4	-15.516667	-3.800000	
52556	52557	12.3	-16.016667	-4.300000	
52557	52558	12.2	-16.516667	-4.800000	
52558	52559	11.9	-17.016667	-5.300000	
52559	52560	11.9	-17.516667	-5.800000	

	demand
48240	509.518789
48241	509.518789
48242	509.518789
48243	509.518789
48244	509.518789
…	…
52555	1149.899894
52556	1146.268346
52557	1013.422674
52558	862.117996

```
52559   776.293497

[4320 rows x 5 columns]
```

- Multi-layer Perceptron (MLP) Regression

[41]:
```
model(pipe_neural, param_neural, X_train, y_train, X, y)
```

```
Fitting 3 folds for each of 72 candidates, totalling 216 fits
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=5,
```

```
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.5s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.1s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.1s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.5s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.2s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
```

```
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.7s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.3s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.1s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.5s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
```

```
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.1s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.2s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.1s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.2s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.2s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.2s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.2s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=5,
```

```
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.5s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.0s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.1s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
```

```
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.1s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.3s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.2s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.9s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.5s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.7s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.0s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.5s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.2s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.4s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
```

```
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.9s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.6s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.0s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.0s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.0s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.9s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.2s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.8s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.2s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.0s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.6s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.8s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   5.0s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.4s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   4.8s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
```

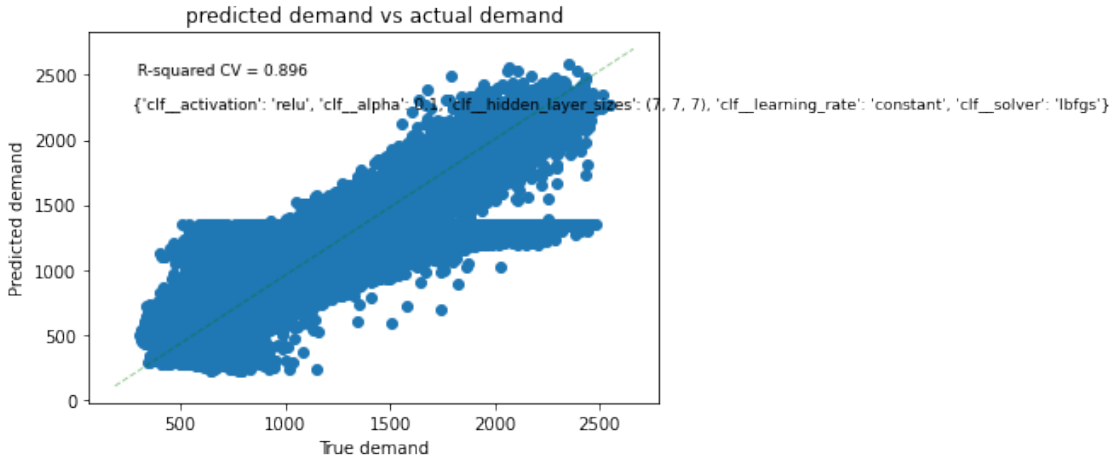
```
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.0s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.5s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.1s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.1s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.0s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   5.7s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   5.0s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.9s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.5s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.9s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
```

```
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.3s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.8s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.3s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.5s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.0s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   5.7s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.9s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.3s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   4.9s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.5s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   5.1s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.9s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.0s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.8s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.5s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.3s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.0s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.6s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.3s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
```

```
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.6s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.8s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   4.8s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.3s
##### Results
    mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0        0.661496      0.096231         0.002961        0.000741
1        0.626830      0.037228         0.002619        0.000404
2        1.873996      0.085555         0.003755        0.000243
3        1.854197      0.033390         0.003557        0.000108
4        2.000484      0.028285         0.003562        0.000153
..            …             …                …               …
67       0.875441      0.055003         0.003407        0.000393
68       3.389447      0.084854         0.004657        0.000117
69       3.326958      0.237724         0.005340        0.000652
70       3.237585      1.164833         0.005781        0.001136
71       2.673600      1.547067         0.005131        0.000247

   param_clf__activation param_clf__alpha param_clf__hidden_layer_sizes  \
0                   relu           0.001                             5
1                   relu           0.001                             5
2                   relu           0.001                        (10, 10)
3                   relu           0.001                        (10, 10)
4                   relu           0.001                       (7, 7, 7)
..                   …               …                             …
67                  tanh             100                             5
68                  tanh             100                        (10, 10)
69                  tanh             100                        (10, 10)
70                  tanh             100                       (7, 7, 7)
71                  tanh             100                       (7, 7, 7)

   param_clf__learning_rate param_clf__solver  \
0                  constant            lbfgs
1                invvscaling            lbfgs
2                  constant            lbfgs
3                invvscaling            lbfgs
4                  constant            lbfgs
..                      …               …
67               invvscaling            lbfgs
68                 constant            lbfgs
69               invvscaling            lbfgs
70                 constant            lbfgs
71               invvscaling            lbfgs
```

```
                                                params  split0_test_score  \
0   {'clf__activation': 'relu', 'clf__alpha': 0.00…           0.828269
1   {'clf__activation': 'relu', 'clf__alpha': 0.00…           0.832614
2   {'clf__activation': 'relu', 'clf__alpha': 0.00…           0.881190
3   {'clf__activation': 'relu', 'clf__alpha': 0.00…           0.904428
4   {'clf__activation': 'relu', 'clf__alpha': 0.00…           0.856931
..                                                  …                …
67  {'clf__activation': 'tanh', 'clf__alpha': 100,…           0.621880
68  {'clf__activation': 'tanh', 'clf__alpha': 100,…           0.676617
69  {'clf__activation': 'tanh', 'clf__alpha': 100,…           0.569742
70  {'clf__activation': 'tanh', 'clf__alpha': 100,…           0.418976
71  {'clf__activation': 'tanh', 'clf__alpha': 100,…           0.584047

    split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0            0.582476           0.576893         0.662546        0.117206
1            0.580846           0.577682         0.663714        0.119437
2            0.903421           0.899403         0.894671        0.009673
3            0.843787           0.901337         0.883184        0.027886
4            0.899812           0.884592         0.880445        0.017750
..                …                  …                …               …
67           0.791307           0.641470         0.684886        0.075675
68           0.715639           0.719473         0.703910        0.019362
69           0.750346           0.582840         0.634309        0.082224
70          -0.000030           0.057074         0.158673        0.185532
71           0.017742          -0.000099         0.200563        0.271262

    rank_test_score
0                44
1                43
2                 5
3                11
4                14
..              …
67               36
68               34
69               48
70               66
71               65

[72 rows x 16 columns]
[500.66671233 465.73227373 462.33966482 … 951.50404044 814.2848072
 671.35693805]
best_index 16
best_score 0.8988070360813705
best_params {'clf__activation': 'relu', 'clf__alpha': 0.1,
'clf__hidden_layer_sizes': (7, 7, 7), 'clf__learning_rate': 'constant',
'clf__solver': 'lbfgs'}
```

```
##### CV Results
mean_score 0.896091361698575
Model does not support model coefficients
Model does not support feature importances
```



	period	temperature	hours before sunrise	hours before sunset	\
48240	48241	11.9	3.833333	20.316667	
48241	48242	12.0	3.333333	19.816667	
48242	48243	12.1	2.833333	19.316667	
48243	48244	12.0	2.333333	18.816667	
48244	48245	11.9	1.833333	18.316667	
…	…	…	…	…	
52555	52556	12.4	-15.516667	-3.800000	
52556	52557	12.3	-16.016667	-4.300000	
52557	52558	12.2	-16.516667	-4.800000	
52558	52559	11.9	-17.016667	-5.300000	
52559	52560	11.9	-17.516667	-5.800000	

	demand
48240	500.666712
48241	465.732274
48242	462.339665
48243	462.339665
48244	462.339665
…	…
52555	1233.554021
52556	1092.529031
52557	951.504040
52558	814.284807
52559	671.356938

[4320 rows x 5 columns]

0.4 Conclusion

KNN fits the best

1. KNN
* Parameters: clf___n_neighbors: 25 * Score: 0.918

2. Decision Tree Regression
* Parameters: clf___max_depth: 10, clf___min_samples_leaf: 10 * Score: 0.911

3. Polynomial Regression
* Parameters: polynomial___degree: 6 * Score: 0.87

4. Random Forest
* Parameters: clf___max_depth: 5, clf___max_features: 2, clf___n_estimators: 50 * Score: 0.883

5. Linear Regression
* Parameters: non * Score: 0.548

6. Lasso
* Parameters: clf___alpha: 0.01 * Score: 0.548

7. Ridge
* Parameters: clf___alpha: 1 * Score: 0.548

8. XGBoost
* Parameters: clf_colsample_bytree: 1, clf_gamma: 0.01, clf_ max_depth: 5, clf_min_child_weight': 6, clf_subsample: 0.7 * Score: 0.918

9. Multi-layer Perceptron (MLP) Regression
* Parameters: 'clf___activation': 'relu', 'clf___alpha': 0.001, 'clf___hidden_layer_sizes': (7, 7, 7), 'clf___learning_rate': 'constant', 'clf___solver': 'lbfgs' * Score: 0.862