Amey

December 12, 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

[1]:
```
!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)
```

[2]:
```
import warnings
warnings.filterwarnings('ignore')
# warnings.filterwarnings(action='once')
```

[15]:
```
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

[16]:
```
file = pd.read_csv('Data.csv')
file = pd.DataFrame(file)
file
```

- Split to train, test and validation datasets

[17]:
```
df = file[file.demand.notnull()]
df
```

[17]:

	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]
```

[18]:
```
y = df.demand
y
```

[18]:
```
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

[19]:
```
X = df.drop('period', axis=1).drop('demand', axis = 1)
X
```

[19]:

	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

[20]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,␣
↪random_state=0)
```

- Validation set

[21]:
```
df_vali = file[file.demand.isnull()]
```

[22]:
```
vali = df_vali.drop('period', axis=1).drop('demand', axis = 1)
vali
```

[22]:

	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

[35]:
```
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

[36]:
```
pipe_ols = Pipeline([('scl', StandardScaler()),
           ('clf', LinearRegression())])

param_ols = {}
```

- XGBoost

[37]:
```
# # - XGBoost

# pipe_xgb = Pipeline([('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

[38]:
```
pipe_knn = Pipeline([('clf', KNeighborsRegressor())])

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

- Lasso

[39]:
```
pipe_lasso = Pipeline([('scl', StandardScaler()),
                       ('clf', Lasso(max_iter=1500))])

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

- Ridge

[40]:
```
pipe_ridge = Pipeline([('scl', StandardScaler()),
                       ('clf', Ridge())])

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

- Polynomial Regression

[41]:
```
pipe_poly = Pipeline([('scl', StandardScaler()),
                      ('polynomial', PolynomialFeatures()),
                      ('clf', LinearRegression())])

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

- Decision Tree Regression

[42]:
```
pipe_tree = Pipeline([('clf', DecisionTreeRegressor())])

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

- Random Forest

[43]:
```
pipe_forest = Pipeline([('clf', RandomForestRegressor())])

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

- MLP Regression

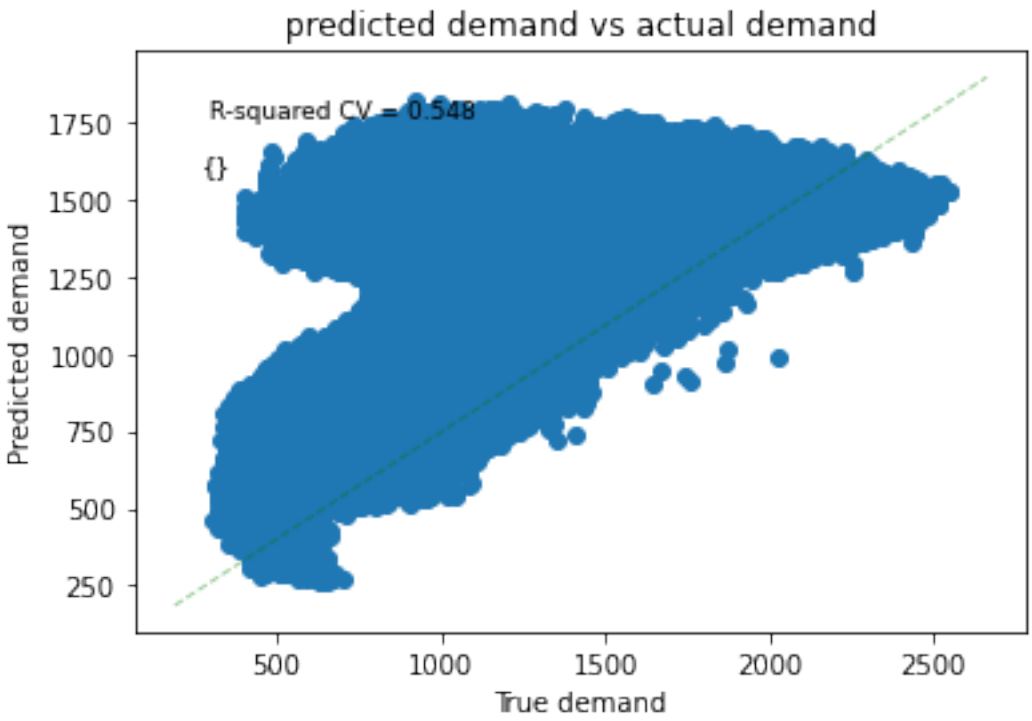
[44]:
```
pipe_neural = Pipeline([('scl', StandardScaler()),
                        ('clf', MLPRegressor())])

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

[45]:
```
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        0.00709      0.000543         0.002471        0.000276     {}

   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

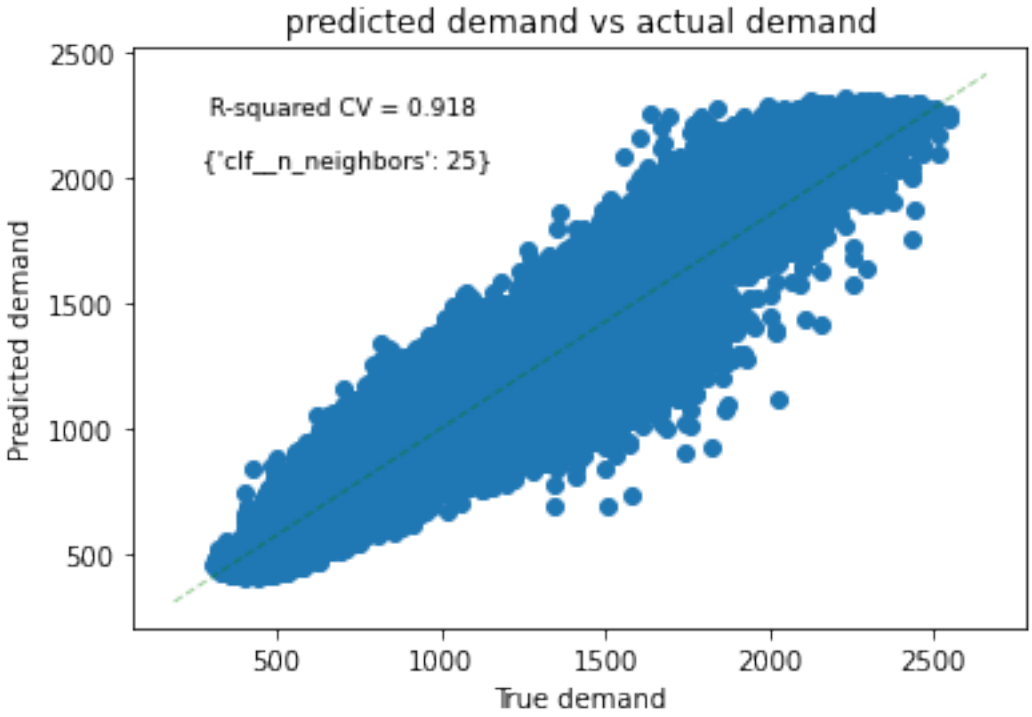
[46]:
```
# model(pipe_xgb, param_xgb, X_train, y_train, X, y)
```

- KNN

[47]:
```
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=   0.0s
[CV] END …clf__n_neighbors=5; total time=   0.0s
[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       0.009826      0.000417         0.038892        0.001207
1       0.010220      0.000744         0.052401        0.000680
2       0.009880      0.000536         0.062440        0.002019
3       0.009335      0.000051         0.082684        0.001527
4       0.009375      0.000065         0.091887        0.002703

  param_clf__n_neighbors                    params  split0_test_score  \
0                      5   {'clf__n_neighbors': 5}           0.909001
1                     10  {'clf__n_neighbors': 10}           0.915740
2                     15  {'clf__n_neighbors': 15}           0.917366
3                     25  {'clf__n_neighbors': 25}           0.918470
4                     30  {'clf__n_neighbors': 30}           0.917932

   split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0           0.905546           0.908140         0.907563        0.001468
1           0.913639           0.914163         0.914514        0.000893
2           0.915896           0.916517         0.916593        0.000602
3           0.916293           0.917259         0.917341        0.000891
4           0.915944           0.917189         0.917022        0.000820
```

```
   rank_test_score
0                5
1                4
2                3
3                1
4                2
[530.16 517.68 487.36 … 846.08 662.6  591.56]
best_index 3
best_score 0.9173405968230345
best_params {'clf__n_neighbors': 25}
##### CV Results
mean_score 0.9175986294004446
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	530.16
48241	48242	12.0	3.333333	19.816667	517.68
48242	48243	12.1	2.833333	19.316667	487.36
48243	48244	12.0	2.333333	18.816667	477.96
48244	48245	11.9	1.833333	18.316667	468.40
…	…	…	…	…	…
52555	52556	12.4	-15.516667	-3.800000	1156.68

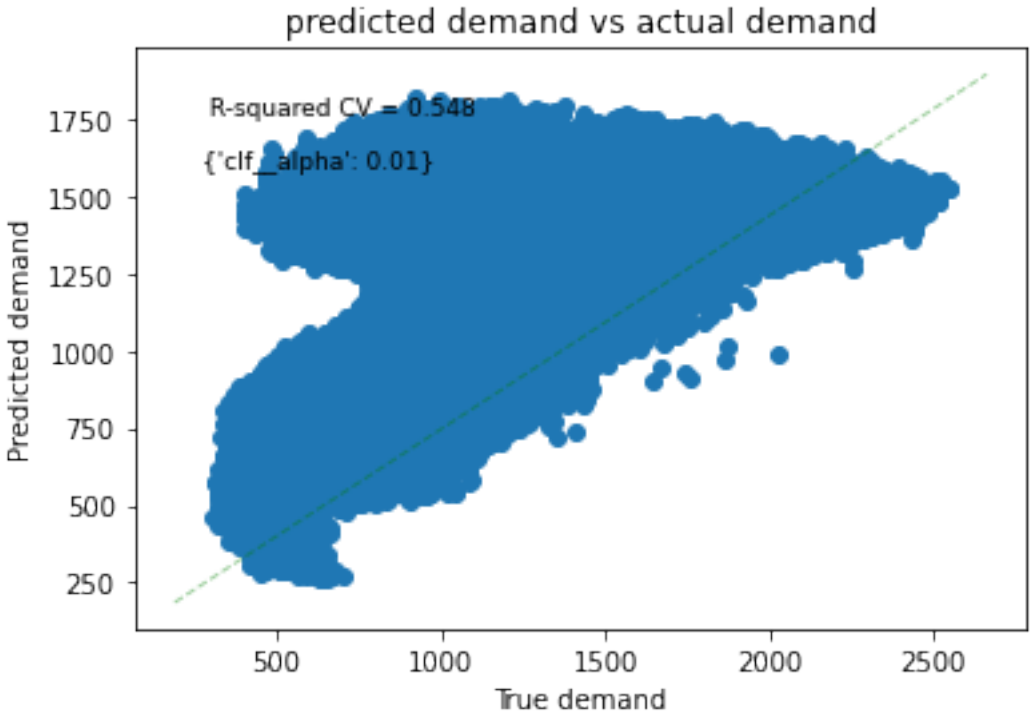
```
52556   52557         12.3           -16.016667           -4.300000  1051.48
52557   52558         12.2           -16.516667           -4.800000   846.08
52558   52559         11.9           -17.016667           -5.300000   662.60
52559   52560         11.9           -17.516667           -5.800000   591.56

[4320 rows x 5 columns]
```

- Lasso

[48]:
```
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       0.011720      0.000095         0.002458        0.000129
1       0.011157      0.000765         0.002509        0.000053
2       0.009200      0.000532         0.002295        0.000022
3       0.006961      0.000784         0.002316        0.000040

  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	

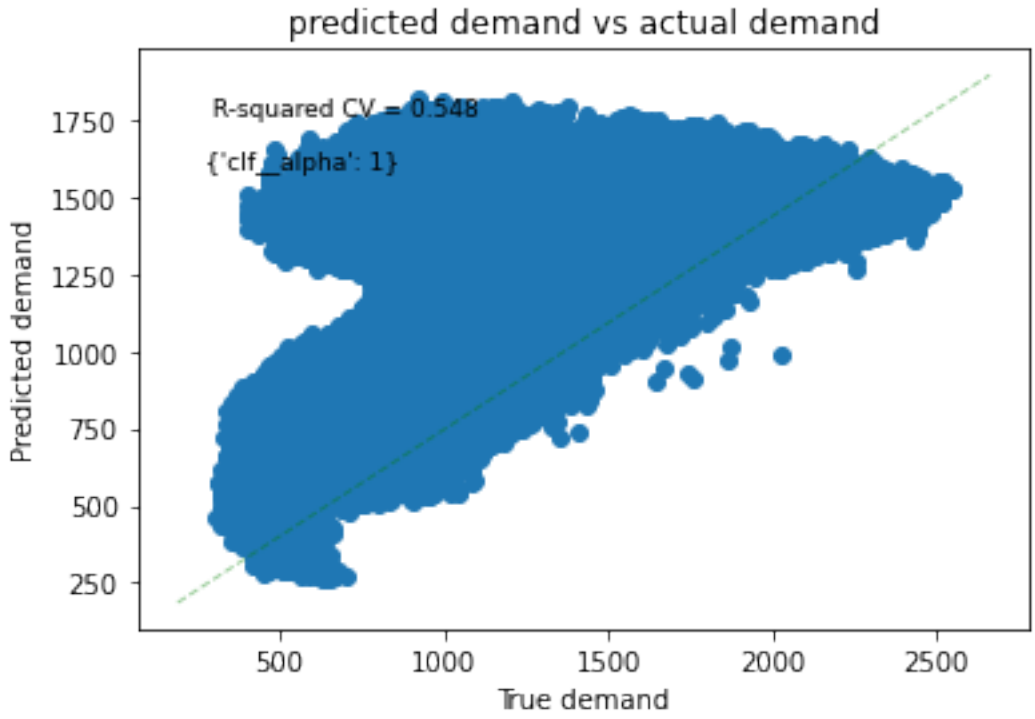
```
            demand
48240   321.529640
48241   342.236281
48242   362.942921
48243   385.439631
48244   407.936342
…              …
52555  1444.061875
52556  1466.558586
52557  1489.055297
52558  1513.342078
52559  1534.943754

[4320 rows x 5 columns]
```

- Ridge

[49]:
```
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       0.006983      0.001807         0.002522        0.000431
1       0.006365      0.001351         0.002424        0.000353
2       0.007045      0.001271         0.002121        0.000306
3       0.005093      0.000227         0.001704        0.000032

  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
```

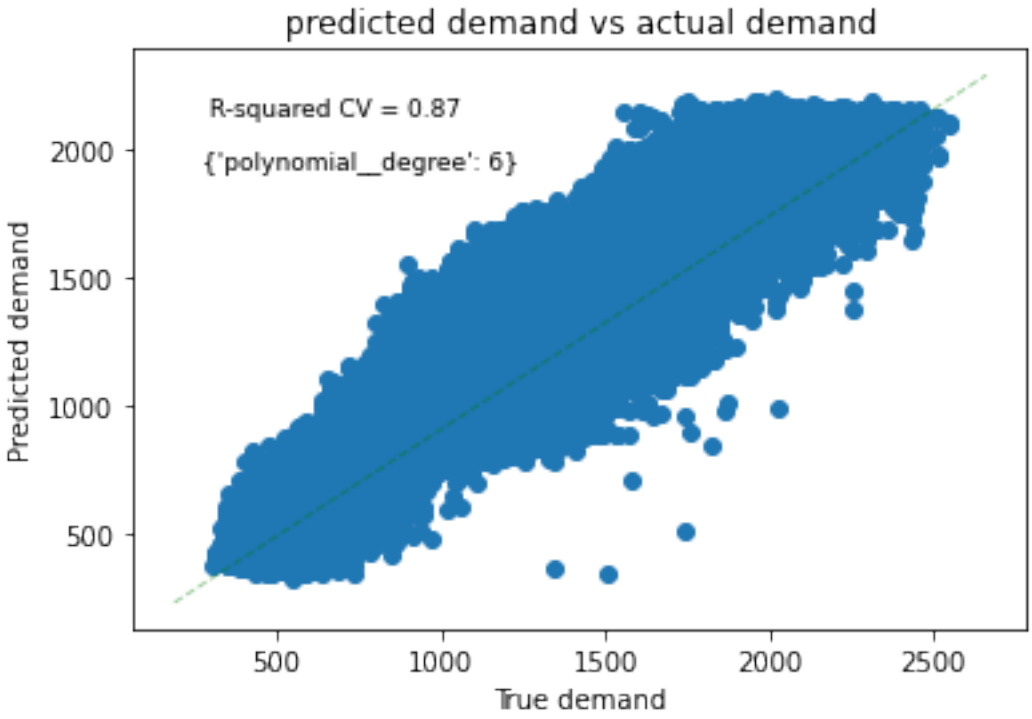
```
48244   48245         11.9              1.833333            18.316667
…       …          …                    …                  …
52555   52556         12.4            -15.516667            -3.800000
52556   52557         12.3            -16.016667            -4.300000
52557   52558         12.2            -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

[50]:
```
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=   0.0s
[CV] END …polynomial__degree=4; total time=   0.0s
[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       0.014971      0.002945         0.004138        0.000539
1       0.036599      0.001038         0.006828        0.000498
2       0.110401      0.005728         0.010378        0.001034

  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

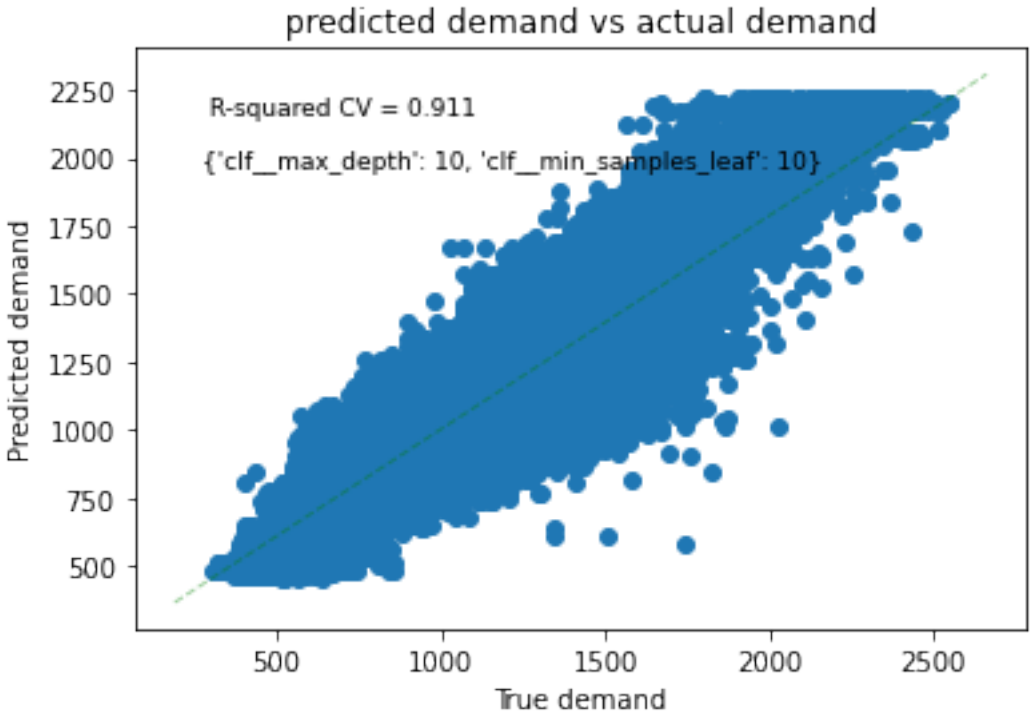
[51]:
```
model(pipe_tree, param_tree, X_train, y_train, X, y)
```

```
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=   0.0s
[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=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        0.011134      0.000591         0.001933        0.000194
1        0.010282      0.000426         0.001859        0.000120
2        0.010123      0.000309         0.001702        0.000056
3        0.010011      0.000217         0.001866        0.000076
4        0.020652      0.000517         0.002883        0.000683
5        0.020783      0.000362         0.002559        0.000708
6        0.020420      0.000293         0.002072        0.000078
7        0.020775      0.000612         0.002293        0.000237
8        0.037138      0.000265         0.002567        0.000155
9        0.038484      0.000864         0.002583        0.000017
10       0.034270      0.001573         0.002573        0.000113
11       0.031923      0.001029         0.002432        0.000038

   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'…          0.907996
9   {'clf__max_depth': 10, 'clf__min_samples_leaf'…          0.909336
10  {'clf__max_depth': 10, 'clf__min_samples_leaf'…          0.906331
11  {'clf__max_depth': 10, 'clf__min_samples_leaf'…          0.897609

    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            0.906542           0.907004         0.907181        0.000606
9            0.907549           0.907204         0.908030        0.000935
10           0.905102           0.904737         0.905390        0.000682
11           0.896899           0.897394         0.897301        0.000297

    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 0.9080296736083269
best_params {'clf__max_depth': 10, 'clf__min_samples_leaf': 10}
##### CV Results
mean_score 0.9110851130156965
Model does not support model coefficients
Feature importances:  [('temperature', 0.049639106318395375), ('hours before
```

```
sunrise', 0.2254406166609969), ('hours before sunset', 0.7249202770206077)]
```



	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

[52]:
```
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=   0.0s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=10; total
time=   0.0s
[CV] END clf__max_depth=1, clf__max_features=None, clf__n_estimators=10; total
time=   0.0s
[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=   0.1s
[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=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=   0.1s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=2, clf__max_features=None, clf__n_estimators=20; total
time=   0.1s
[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=   0.0s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=10; total
time=   0.0s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=10; total
time=   0.0s
[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=   0.2s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=50; total
time=   0.2s
[CV] END clf__max_depth=2, clf__max_features=1, clf__n_estimators=50; total
time=   0.2s
[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
```

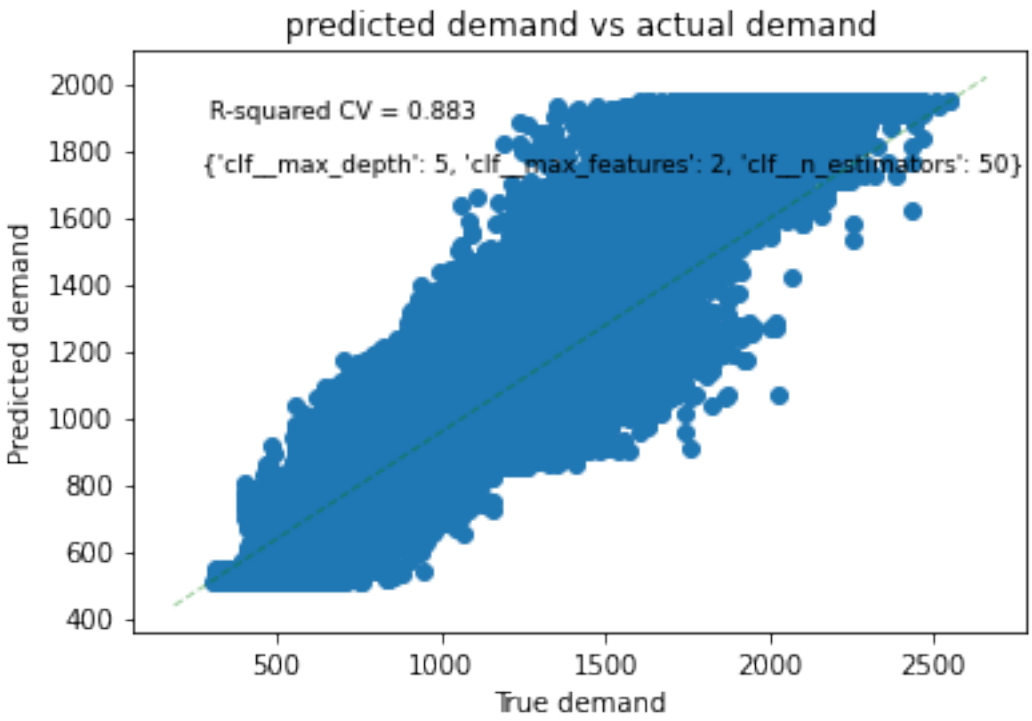
```
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=20; total
time=   0.1s
[CV] END clf__max_depth=5, clf__max_features=1, clf__n_estimators=50; total
time=   0.4s
[CV] END clf__max_depth=5, 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=   0.6s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=50; total
time=   0.6s
[CV] END clf__max_depth=5, clf__max_features=2, clf__n_estimators=50; total
time=   0.6s
##### Results
    mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0        0.043469      0.001117         0.004038        0.000080
1        0.081017      0.001199         0.006647        0.000657
2        0.207130      0.014803         0.014670        0.002282
3        0.032488      0.000681         0.004917        0.000344
4        0.055928      0.001697         0.007117        0.001071
5        0.130357      0.000924         0.013099        0.000121
6        0.037700      0.000903         0.004355        0.000388
7        0.071457      0.002586         0.006725        0.000156
8        0.178148      0.004057         0.014855        0.002646
9        0.068519      0.000558         0.005609        0.000515
10       0.139546      0.000575         0.008218        0.000451
11       0.336381      0.001428         0.016573        0.000542
12       0.039052      0.001773         0.004993        0.000096
13       0.076120      0.000994         0.007939        0.000334
14       0.188770      0.004812         0.017353        0.000791
15       0.058623      0.002477         0.005386        0.000376
16       0.109966      0.003355         0.009021        0.000288
```

```
17       0.256334      0.005952         0.016754        0.000893
18       0.138388      0.002347         0.006782        0.000059
19       0.274116      0.003119         0.011248        0.000087
20       0.667539      0.002830         0.025380        0.000221
21       0.070038      0.001236         0.007524        0.000711
22       0.136199      0.002251         0.011544        0.000192
23       0.334877      0.003614         0.025403        0.000575
24       0.104141      0.001957         0.007346        0.000586
25       0.215473      0.004304         0.013363        0.001382
26       0.539003      0.002780         0.026809        0.000914

   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…           0.409458
1   {'clf__max_depth': 1, 'clf__max_features': Non…           0.427587
2   {'clf__max_depth': 1, 'clf__max_features': Non…           0.413499
3   {'clf__max_depth': 1, 'clf__max_features': 1, …           0.424206
4   {'clf__max_depth': 1, 'clf__max_features': 1, …           0.321034
5   {'clf__max_depth': 1, 'clf__max_features': 1, …           0.396814
6   {'clf__max_depth': 1, 'clf__max_features': 2, …           0.461713
```

7	{'clf__max_depth': 1, 'clf__max_features': 2, …	0.457805
8	{'clf__max_depth': 1, 'clf__max_features': 2, …	0.459235
9	{'clf__max_depth': 2, 'clf__max_features': Non…	0.622426
10	{'clf__max_depth': 2, 'clf__max_features': Non…	0.630156
11	{'clf__max_depth': 2, 'clf__max_features': Non…	0.629646
12	{'clf__max_depth': 2, 'clf__max_features': 1, …	0.472733
13	{'clf__max_depth': 2, 'clf__max_features': 1, …	0.580846
14	{'clf__max_depth': 2, 'clf__max_features': 1, …	0.586879
15	{'clf__max_depth': 2, 'clf__max_features': 2, …	0.658512
16	{'clf__max_depth': 2, 'clf__max_features': 2, …	0.661222
17	{'clf__max_depth': 2, 'clf__max_features': 2, …	0.661863
18	{'clf__max_depth': 5, 'clf__max_features': Non…	0.879036
19	{'clf__max_depth': 5, 'clf__max_features': Non…	0.879085
20	{'clf__max_depth': 5, 'clf__max_features': Non…	0.878044
21	{'clf__max_depth': 5, 'clf__max_features': 1, …	0.798301
22	{'clf__max_depth': 5, 'clf__max_features': 1, …	0.847905
23	{'clf__max_depth': 5, 'clf__max_features': 1, …	0.830141
24	{'clf__max_depth': 5, 'clf__max_features': 2, …	0.879659
25	{'clf__max_depth': 5, 'clf__max_features': 2, …	0.881775
26	{'clf__max_depth': 5, 'clf__max_features': 2, …	0.883930

	split1_test_score	split2_test_score	mean_test_score	std_test_score	\
0	0.405206	0.422511	0.412392	0.007363	
1	0.409134	0.417465	0.418062	0.007545	
2	0.410072	0.418733	0.414101	0.003561	
3	0.394326	0.335597	0.384710	0.036808	
4	0.402534	0.370365	0.364645	0.033517	
5	0.372575	0.414443	0.394611	0.017163	
6	0.465179	0.471537	0.466143	0.004068	
7	0.456934	0.470286	0.461675	0.006099	
8	0.457877	0.467334	0.461482	0.004175	
9	0.613380	0.624200	0.620002	0.004738	
10	0.610882	0.627085	0.622708	0.008456	
11	0.615067	0.632286	0.625666	0.007572	
12	0.625005	0.511639	0.536459	0.064595	
13	0.607795	0.592538	0.593726	0.011034	
14	0.597784	0.597701	0.594121	0.005121	
15	0.644056	0.654388	0.652319	0.006080	
16	0.648701	0.661102	0.657008	0.005875	
17	0.652278	0.669181	0.661107	0.006921	
18	0.872339	0.876674	0.876016	0.002773	
19	0.873873	0.878721	0.877226	0.002376	
20	0.875601	0.878667	0.877438	0.001323	
21	0.803076	0.807701	0.803026	0.003837	
22	0.821741	0.811561	0.827069	0.015308	
23	0.827082	0.828521	0.828582	0.001249	
24	0.882614	0.874783	0.879019	0.003229	
25	0.875167	0.885630	0.880857	0.004321	

```
26           0.881778           0.883998         0.883235        0.001031

    rank_test_score
0                24
1                22
2                23
3                26
4                27
5                25
6                19
7                20
8                21
9                15
10               14
11               13
12               18
13               17
14               16
15               12
16               11
17               10
18                6
19                5
20                4
21                9
22                8
23                7
24                3
25                2
26                1
[ 508.04086737  508.04086737  508.04086737 … 1051.37809806  867.34975045
  752.3030571 ]
best_index 26
best_score 0.8832349910787602
best_params {'clf__max_depth': 5, 'clf__max_features': 2, 'clf__n_estimators':
50}
##### CV Results
mean_score 0.8826708441241793
Model does not support model coefficients
Feature importances:  [('temperature', 0.07674538201360276), ('hours before
sunrise', 0.34922074801091474), ('hours before sunset', 0.5740338699754826)]
```



	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	508.040867
48241	508.040867
48242	508.040867
48243	508.040867
48244	508.040867
…	…
52555	1188.241934
52556	1172.252128
52557	1051.378098
52558	867.349750

```
52559   752.303057

[4320 rows x 5 columns]
```

- Multi-layer Perceptron (MLP) Regression

[53]:
```
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=   0.5s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.5s
[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=invscaling, clf__solver=lbfgs; total time=   0.2s
[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=   1.6s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.7s
[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=   1.7s
[CV] END clf__activation=relu, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.7s
[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=(7, 7,
7), clf__learning_rate=constant, 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=   2.2s
[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=   2.2s
[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=   2.2s
[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=   1.9s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.5s
[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=   0.7s
[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=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=(10,
10), clf__learning_rate=constant, 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=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=   2.0s
[CV] END clf__activation=relu, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.2s
[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=   1.7s
[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=   1.8s
[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=   1.8s
[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=0.01, 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=0.01, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[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=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=   0.6s
[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=(10, 10),
clf__learning_rate=constant, 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=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=   1.8s
[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=(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=   2.1s
[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=constant, clf__solver=lbfgs; total time=   2.4s
[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=   2.5s
[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=   2.3s
[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=   2.5s
[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=   0.7s
[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=invscaling, clf__solver=lbfgs; total time=   0.3s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.4s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.6s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.7s
[CV] END clf__activation=relu, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.7s
[CV] END clf__activation=relu, clf__alpha=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=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=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.7s
[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=   1.9s
[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=   1.8s
[CV] END clf__activation=relu, clf__alpha=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=1, clf__hidden_layer_sizes=(7, 7, 7),
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=invscaling, clf__solver=lbfgs; total time=   1.9s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.7s
[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=   0.5s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.3s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.2s
[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=constant, clf__solver=lbfgs; total time=   1.7s
[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=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=   1.8s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[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=   2.2s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.1s
[CV] END clf__activation=relu, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=relu, clf__alpha=10, 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=10, 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=100, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.6s
[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=   0.6s
[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=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=   1.6s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.7s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.6s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.6s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.6s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.6s
[CV] END clf__activation=relu, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.8s
[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=   1.9s
[CV] END clf__activation=relu, 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=relu, 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=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=   0.8s
[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=   1.0s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.4s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.0s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.2s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.9s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
```

```
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.1s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.1s
[CV] END clf__activation=tanh, clf__alpha=0.001, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.6s
[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=   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=   3.8s
[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=   4.0s
[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=   1.2s
[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=   0.9s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   0.9s
[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=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.0s
[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=(10,
10), clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.9s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=constant, 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=constant, 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=   3.0s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=0.01, clf__hidden_layer_sizes=(10,
10), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   4.0s
[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=   0.3s
[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=   4.8s
[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=   3.5s
[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=   4.5s
[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=   0.8s
[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=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=   0.8s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.5s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.2s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.1s
[CV] END clf__activation=tanh, clf__alpha=0.1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.7s
[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=(10, 10),
clf__learning_rate=invscaling, 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=invscaling, clf__solver=lbfgs; total time=   2.8s
[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=   0.1s
[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=   4.7s
[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=   0.4s
[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=   5.3s
[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=   0.5s
[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=   5.5s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.0s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
```

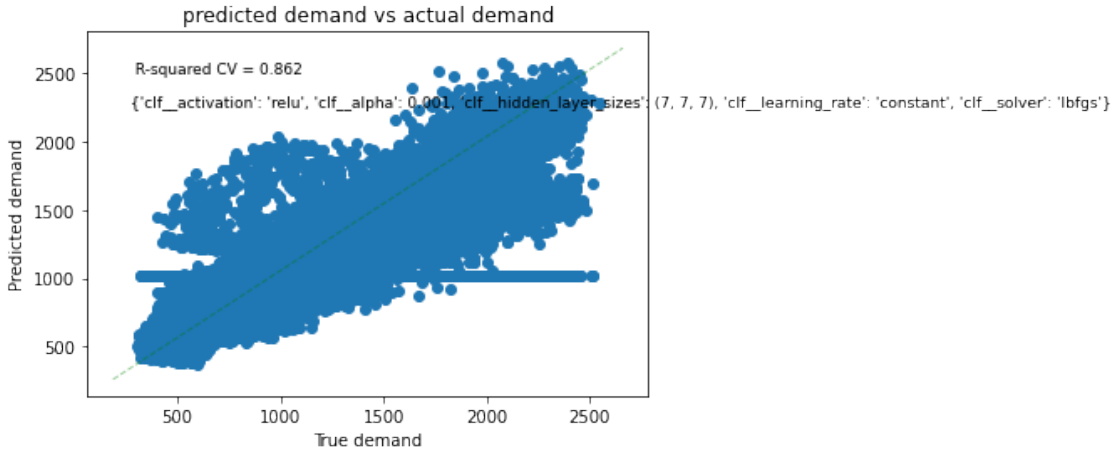
```
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.2s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.1s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, 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=   1.0s
[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=   3.3s
[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=   3.1s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.5s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.7s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=tanh, clf__alpha=1, 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=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.4s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.9s
[CV] END clf__activation=tanh, clf__alpha=1, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.3s
[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=   0.1s
[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=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=   1.0s
[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=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.1s
[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=   3.1s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   3.2s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.9s
[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=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.6s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   4.6s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   1.1s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   5.5s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   1.8s
[CV] END clf__activation=tanh, clf__alpha=10, clf__hidden_layer_sizes=(7, 7, 7),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   4.7s
[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=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=   0.9s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=5,
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   0.4s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   2.8s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=constant, clf__solver=lbfgs; total time=   3.4s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(10, 10),
clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   2.8s
[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=   2.9s
[CV] END clf__activation=tanh, clf__alpha=100, 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=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=constant, clf__solver=lbfgs; total time=   4.0s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   4.3s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   5.3s
[CV] END clf__activation=tanh, clf__alpha=100, clf__hidden_layer_sizes=(7, 7,
7), clf__learning_rate=invscaling, clf__solver=lbfgs; total time=   5.0s
##### Results
    mean_fit_time  std_fit_time  mean_score_time  std_score_time  \
0        0.554897      0.020260         0.003289        0.000616
1        0.454644      0.173347         0.002676        0.000400
2        1.694442      0.104064         0.003267        0.000093
3        1.759638      0.060039         0.003641        0.000214
4        2.033563      0.090458         0.003836        0.000625
..            …             …                …               …
67       0.742136      0.210942         0.003533        0.000382
68       3.210097      0.282265         0.005056        0.000679
69       3.032816      0.291888         0.004489        0.000313
70       2.822582      1.543837         0.005725        0.000546
71       4.852538      0.392508         0.006293        0.000469

   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…           0.831908
1   {'clf__activation': 'relu', 'clf__alpha': 0.00…           0.574359
2   {'clf__activation': 'relu', 'clf__alpha': 0.00…           0.894616
3   {'clf__activation': 'relu', 'clf__alpha': 0.00…           0.894166
4   {'clf__activation': 'relu', 'clf__alpha': 0.00…           0.895319
..                                                 …                …
67  {'clf__activation': 'tanh', 'clf__alpha': 100,…           0.641083
68  {'clf__activation': 'tanh', 'clf__alpha': 100,…           0.632171
69  {'clf__activation': 'tanh', 'clf__alpha': 100,…           0.580067
70  {'clf__activation': 'tanh', 'clf__alpha': 100,…          -0.000020
71  {'clf__activation': 'tanh', 'clf__alpha': 100,…           0.006785

    split1_test_score  split2_test_score  mean_test_score  std_test_score  \
0            0.818977           0.850786         0.833890        0.013061
1            0.774649           0.831894         0.726968        0.110412
2            0.897433           0.892455         0.894834        0.002038
3            0.895533           0.902007         0.897235        0.003420
4            0.899472           0.898038         0.897610        0.001722
..                …                  …                …               …
67           0.645847           0.564585         0.617172        0.037235
68           0.724033           0.752965         0.703056        0.051496
69           0.589566           0.507371         0.559002        0.036714
70           0.552069           0.473564         0.341871        0.243869
71           0.452576           0.391018         0.283460        0.197246

    rank_test_score
0                20
1                31
2                 5
3                 3
4                 1
..              …
67               50
68               32
69               55
70               62
71               67

[72 rows x 16 columns]
[537.0052627  521.63028325 506.25530381 … 938.2405219  810.57541918
 676.93547165]
best_index 4
best_score 0.8976099133817449
best_params {'clf__activation': 'relu', 'clf__alpha': 0.001,
'clf__hidden_layer_sizes': (7, 7, 7), 'clf__learning_rate': 'constant',
'clf__solver': 'lbfgs'}
```

```
##### CV Results
mean_score 0.8620766955405965
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	537.005263
48241	521.630283
48242	506.255304
48243	491.621076
48244	476.986848
…	…
52555	1201.537187
52556	1069.888854
52557	938.240522
52558	810.575419
52559	676.935472

[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