KNN Assignment1

Aakash Ahuja

20/07/2020

```
#tinytex::install_tinytex()
```

```
library(reticulate)
use_python('/Users/aakash/opt/anaconda3/bin/python3.7', required = T)

#conda_create("r-reticulate")
use_condaenv("r-reticulate", required=T)
#conda_install("r-reticulate","seaborn",pip = T)
#conda_install("r-reticulate","PyQt5",pip = T)
```

Loading the libraries

```
from sklearn import datasets
import numpy as np
import pandas as pd

from sklearn import datasets
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, GridSearchCV, PredefinedSplit
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from datetime import datetime

import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns

import matplotlib.pyplot as plt
import PyQt5
from sklearn.datasets import load_digits
from __future__ import print_function

import _pickle as cPickle
import tarfile
from sklearn.linear_model import SGDClassifier
```

Loading the DIgits Dataset

```
# Loading the Digits dataset
digits = load_digits()

# Print to show there are 1797 images (8 by 8 images for a dimensionality of 64)
print(f"Image Data Shape {digits.data.shape}")

# Print to show there are 1797 labels (integers from 0-9)
```

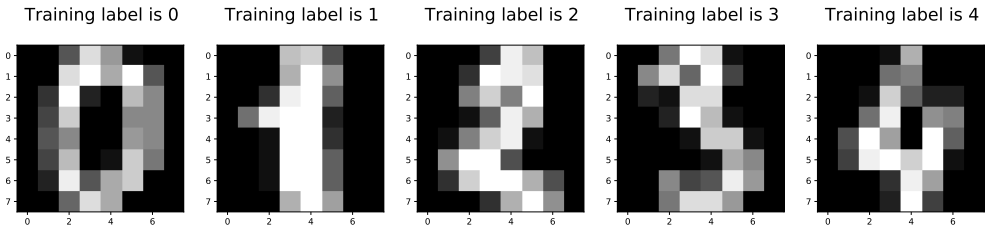
```
## Image Data Shape (1797, 64)
```

```
print(f"Label Data Shape {digits.target.shape}")
```

```
## Label Data Shape (1797,)
```

Showing the Images and the Labels (Digits Dataset)

```
plt.figure(figsize=(20,4))
for index, (image, label) in enumerate(zip(digits.data[0:5], digits.target[0:5])):
    plt.subplot(1, 5, index + 1)
    plt.imshow(np.reshape(image, (8,8)), cmap=plt.cm.gray)
    plt.title('Training label is %i\n' % label, fontsize = 20)
```



Splitting Data into Training and Test Sets (Digits Dataset)

We make training and test sets to make sure that after we train our classification algorithm, it is able to generalize well to new data (we want to test our trained model objectively).

```
# Loading the Digits dataset
digits = load_digits()

X_train, X_test, y_train, y_test = train_test_split(digits.data, digits.target, test_size=0.25, random_s
print("Train size:", X_train.shape[0])
```

```
## Train size: 1347
```

```
print("Test size:", X_test.shape[0])
```

```
## Test size: 450
```

```
print(digits.data.shape)
```

```
## (1797, 64)
```

```
experimentLog = pd.DataFrame(columns=["Model", "Dataset", "TrainAcc", "TestAcc", "TrainTime(sec)", "Tes
```

```
#Set style for plotting
sns.set(style="whitegrid", font_scale=1.3)
matplotlib.rcParams["legend.framealpha"] = 1
matplotlib.rcParams["legend.frameon"] = True

knn_sk = KNeighborsClassifier(n_jobs=-1)

n_neighbors_range = list(range(1,6))
p_range = list (range(1,4))
parameters = {'n_neighbors': n_neighbors_range, 'p': p_range}

np.random.seed(42) # for multiple runs of the same model training, you should re-execute this line
scores = ['accuracy']  # limited to accuracy for now but could use other metrics such as precision, rec
start_time = datetime.now()

for score in scores:
    print("# Tuning hyper-parameters for %s" % score)
    print()

    gridSearch = GridSearchCV(estimator=knn_sk,
                          param_grid=parameters,
                          cv=5,
                          #refit=False,
                          return_train_score=False,
                          verbose=1,
                          n_jobs=-1  #use multiple CPUs; divide and conquer!
                          )
    gridSearch.fit(X_train, y_train)
    print("Best parameters set found on development set:")
    print()
    print(gridSearch.best_params_)
    print()
    print("Grid scores on development set:")
    print()
    print(f'{score}  (+/-stdev*2)     hyper-params')

    means = gridSearch.cv_results_['mean_test_score']  #access elements of the grid search results dict
    stds = gridSearch.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, gridSearch.cv_results_['params']):
        print("%0.3f (+/-%0.03f) for %r"
              % (mean, std * 2, params))
    print()
    end_time = datetime.now()
    wallTimeInSecondsTrain = (end_time - start_time).total_seconds()

    print("Detailed classification report:")
    print()
```

```
    print("The model is trained on the full development set.")
    print("The scores are computed on the full evaluation set.")
    print()
    start_time = datetime.now()
    y_true, y_pred = y_test, gridSearch.predict(X_test)
    end_time = datetime.now()
    wallTimeInSecondsTest = (end_time - start_time).total_seconds()
    print(classification_report(y_true, y_pred))  # more detailed breakdown of the test perf; optional
    print()
    trainAcc = gridSearch.best_score_ #CV accuracy score for best hyperparameter combo

    testAcc  = accuracy_score(y_true, y_pred)
    experimentLog.loc[len(experimentLog)] =["knn", "Digits", f"{trainAcc*100:8.2f}%",
    f"{testAcc*100:8.2f}%",
    f"{wallTimeInSecondsTrain:8.2f} secs", f"{wallTimeInSecondsTest:8.2f} secs",
    f"{gridSearch.best_params_}","5-foldCV-based gridSearch BEST model"]
```

```
## # Tuning hyper-parameters for accuracy
##
## Fitting 5 folds for each of 15 candidates, totalling 75 fits
## GridSearchCV(cv=5, estimator=KNeighborsClassifier(n_jobs=-1), n_jobs=-1,
##              param_grid={'n_neighbors': [1, 2, 3, 4, 5], 'p': [1, 2, 3]},
##              verbose=1)
## Best parameters set found on development set:
##
## {'n_neighbors': 3, 'p': 2}
##
## Grid scores on development set:
##
## accuracy  (+/-stdev*2)     hyper-params
## 0.978 (+/-0.005) for {'n_neighbors': 1, 'p': 1}
## 0.984 (+/-0.014) for {'n_neighbors': 1, 'p': 2}
## 0.984 (+/-0.007) for {'n_neighbors': 1, 'p': 3}
## 0.973 (+/-0.009) for {'n_neighbors': 2, 'p': 1}
## 0.979 (+/-0.011) for {'n_neighbors': 2, 'p': 2}
## 0.980 (+/-0.009) for {'n_neighbors': 2, 'p': 3}
## 0.980 (+/-0.010) for {'n_neighbors': 3, 'p': 1}
## 0.986 (+/-0.003) for {'n_neighbors': 3, 'p': 2}
## 0.986 (+/-0.007) for {'n_neighbors': 3, 'p': 3}
## 0.978 (+/-0.009) for {'n_neighbors': 4, 'p': 1}
## 0.984 (+/-0.013) for {'n_neighbors': 4, 'p': 2}
## 0.984 (+/-0.012) for {'n_neighbors': 4, 'p': 3}
## 0.981 (+/-0.015) for {'n_neighbors': 5, 'p': 1}
## 0.986 (+/-0.014) for {'n_neighbors': 5, 'p': 2}
## 0.984 (+/-0.013) for {'n_neighbors': 5, 'p': 3}
##
## Detailed classification report:
##
## The model is trained on the full development set.
## The scores are computed on the full evaluation set.
##
##               precision    recall  f1-score   support
##
```

```
##            0       1.00      1.00      1.00        37
##            1       1.00      0.98      0.99        43
##            2       0.98      1.00      0.99        44
##            3       0.96      0.98      0.97        45
##            4       1.00      0.97      0.99        38
##            5       0.98      0.98      0.98        48
##            6       1.00      1.00      1.00        52
##            7       0.98      1.00      0.99        48
##            8       1.00      0.96      0.98        48
##            9       0.98      1.00      0.99        47
##
##     accuracy                           0.99       450
##    macro avg       0.99      0.99      0.99       450
## weighted avg       0.99      0.99      0.99       450
##
##
##
## [Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
## [Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:    4.9s
## [Parallel(n_jobs=-1)]: Done  75 out of  75 | elapsed:    7.2s finished
```

```
experiments=pd.DataFrame(experimentLog)
```

```
print(py$experimentLog)
```

```
##   Model Dataset  TrainAcc   TestAcc TrainTime(sec) TestTime(sec)
## 0   knn  Digits    98.59%    98.67%      7.24 secs     0.06 secs
##                        Param                            Description
## 0 {'n_neighbors': 3, 'p': 2} 5-foldCV-based gridSearch BEST model
```

```
best_p = gridSearch.best_params_["p"]
best_n = gridSearch.best_params_["n_neighbors"]
print("best_p: ",best_p)
```

```
## best_p:  2
```

```
print("best_n: ",best_n)
```

```
## best_n:  3
```

```
# have a look at CV dictionary of results
gridSearch.cv_results_
```

```
## {'mean_fit_time': array([0.00735459, 0.00948186, 0.04355631, 0.00515003, 0.00911751,
##        0.0608932 , 0.01440153, 0.00770197, 0.0248292 , 0.01453419,
##        0.01428933, 0.01824551, 0.00690842, 0.00799465, 0.04188929]), 'std_fit_time': array([0.002098
##        0.05175832, 0.01234919, 0.00475734, 0.02194056, 0.01664058,
##        0.00586322, 0.02028844, 0.00208729, 0.00528395, 0.0491504 ]), 'mean_score_time': array([0.078
##        0.5006938 , 0.06209502, 0.09827204, 0.49329896, 0.08459706,
##        0.14080501, 0.52171597, 0.07889695, 0.08143115, 0.48648124]), 'std_score_time': array([0.0210
```

```
##        0.05850746, 0.02019864, 0.04614617, 0.0772659 , 0.02072508,
##        0.04853649, 0.06322336, 0.02155887, 0.0204447 , 0.11481141]), 'param_n_neighbors': masked_arra
##             mask=[False, False, False, False, False, False, False, False,
##                   False, False, False, False, False, False, False, False],
##        fill_value='?',
##             dtype=object), 'param_p': masked_array(data=[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3
##             mask=[False, False, False, False, False, False, False, False,
##                   False, False, False, False, False, False, False, False],
##        fill_value='?',
##             dtype=object), 'params': [{'n_neighbors': 1, 'p': 1}, {'n_neighbors': 1, 'p': 2}, {'n_nei
##        0.97407407, 0.97777778, 0.98518519, 0.98518519, 0.97407407,
##        0.98148148, 0.98518519, 0.98148148, 0.98888889, 0.98518519]), 'split1_test_score': array([0.97
##        0.97777778, 0.98888889, 0.98888889, 0.99259259, 0.98518519,
##        0.9962963 , 0.99259259, 0.99259259, 0.9962963 , 0.99259259]), 'split2_test_score': array([0.97
##        0.98513011, 0.98141264, 0.98513011, 0.98141264, 0.97769517,
##        0.97769517, 0.97769517, 0.97026022, 0.9739777 , 0.9739777 ]), 'split3_test_score': array([0.98
##        0.97769517, 0.97769517, 0.98513011, 0.98513011, 0.9739777 ,
##        0.98141264, 0.97769517, 0.97769517, 0.98513011, 0.98141264]), 'split4_test_score': array([0.97
##        0.98513011, 0.9739777 , 0.98513011, 0.98513011, 0.98141264,
##        0.98141264, 0.98884758, 0.98513011, 0.98513011, 0.98884758]), 'mean_test_score': array([0.977
##        0.97996145, 0.97995043, 0.98589288, 0.98589013, 0.97846895,
##        0.98365964, 0.98440314, 0.98143192, 0.98588462, 0.98440314]), 'std_test_score': array([0.00235
##        0.00442709, 0.00505004, 0.00149816, 0.00365029, 0.00433262,
##        0.00648232, 0.00595694, 0.00744325, 0.00721584, 0.00640414]), 'rank_test_score': array([14,  4
##       dtype=int32)}
```

Visualize the performance metrics along each hyperparameter

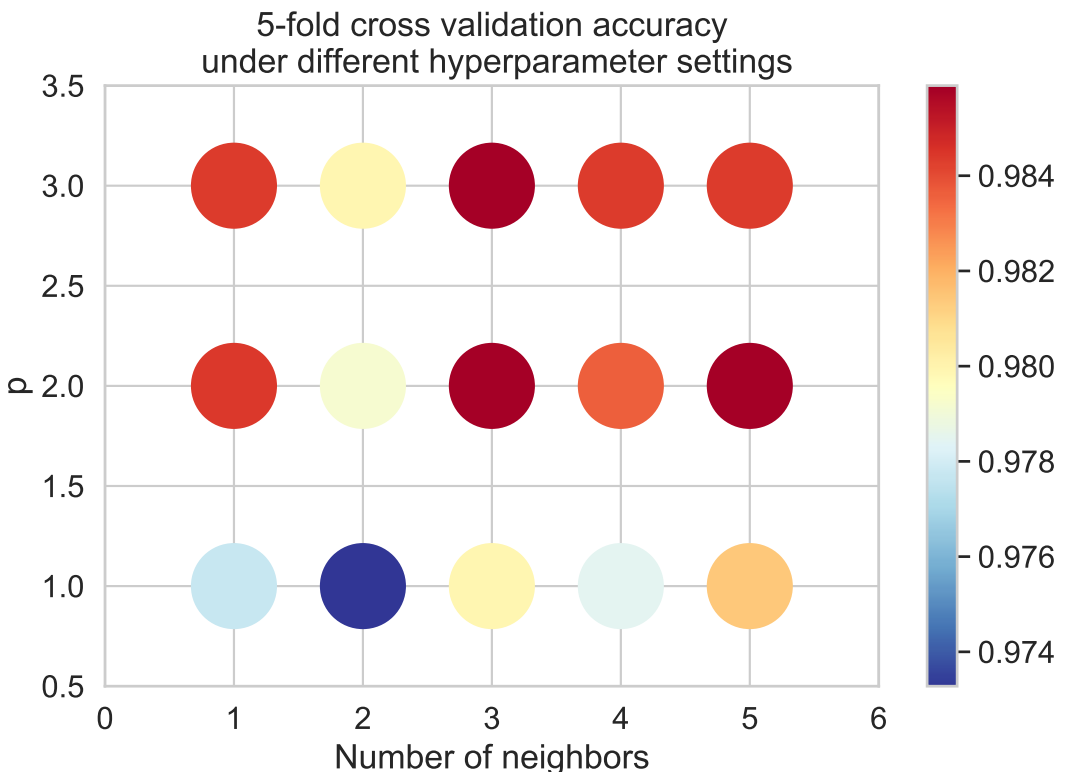
```
plt.figure(figsize=(8, 5))
sc = plt.scatter(gridSearch.cv_results_["param_n_neighbors"],
                 gridSearch.cv_results_["param_p"],
                 c=gridSearch.cv_results_["mean_test_score"],
                 cmap = plt.get_cmap("RdYlBu_r"),
                 s=1500)
plt.colorbar(sc)
```

```
## <matplotlib.colorbar.Colorbar object at 0x122d006d0>
```

```
plt.grid(True)
plt.gca().set_axisbelow(True)
plt.xlabel("Number of neighbors")
plt.ylabel("p")
plt.title("5-fold cross validation accuracy\n under different hyperparameter settings")
plt.xlim([0, 6])
```

```
## (0.0, 6.0)
```

```
plt.ylim([0.5, 3.5]);
plt.show()
```



Let's choose the best p^* and see how accuracy depends on the number of nearest neightbors for a KNN classifier p

```
p_idx = gridSearch.cv_results_["param_p"].data == best_p
plt.figure(figsize=(8,6))
plt.plot(gridSearch.cv_results_["param_n_neighbors"].data[p_idx],
         gridSearch.cv_results_["mean_test_score"][p_idx],
         "o-")
plt.plot([best_n], [gridSearch.cv_results_["mean_test_score"][p_idx][best_n - 1]], "or", markersize=15,
plt.xlabel("Number of neighbors")
plt.ylabel("Accuracy")
plt.title("")
plt.grid(True)
plt.legend(numpoints=1)
plt.show()
```

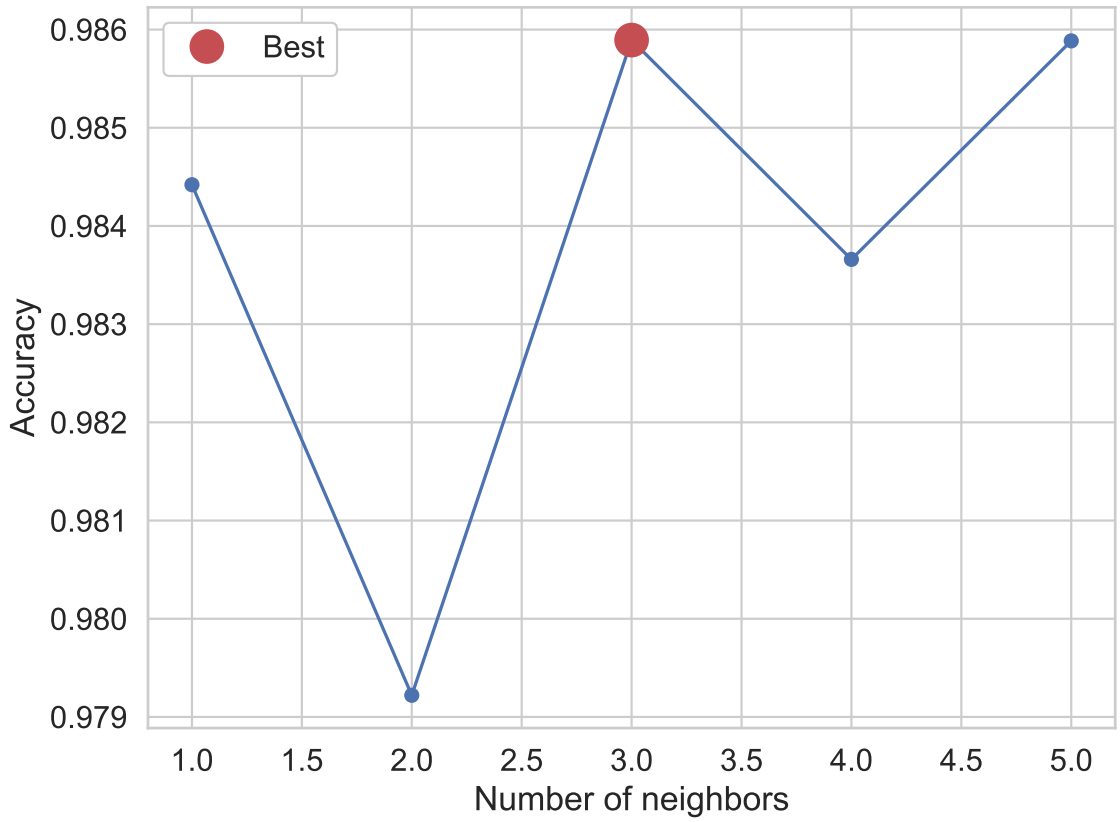


Image classification dataset: CIFAR10

Upacking the Data and doing EDA

In this section we unpack the CIFAR-10 Data set and do some exploratory data analysis

```
sns.set(style="whitegrid", font_scale=1.3)
matplotlib.rcParams["legend.framealpha"] = 1
matplotlib.rcParams["legend.frameon"] = True
np.random.seed(42)


def unpickle(file):
    fo = open(file, 'rb')
    dict = cPickle.load(fo, encoding='latin1')
    fo.close()
    return dict


#/root/tmp/cifar-10-batches-py/data_batch_
for b in range(1, 6):
```

```
    data_batch = unpickle("/Users/aakash/Desktop/IUB/AML/I526_AML_SP20-master/Assignments/Unit-02_KNN_C
    if b == 1:
        X_train = data_batch["data"]
        y_train = np.array(data_batch["labels"])
    else:
        X_train = np.append(X_train, data_batch["data"], axis=0)
        y_train = np.append(y_train, data_batch["labels"], axis=0)

data_batch = unpickle("/Users/aakash/Desktop/IUB/AML/I526_AML_SP20-master/Assignments/Unit-02_KNN_CIFAR
X_test = data_batch["data"]
y_test = np.array(data_batch["labels"])

#Read meta-information file with the names of the classes
classes = unpickle("/Users/aakash/Desktop/IUB/AML/I526_AML_SP20-master/Assignments/Unit-02_KNN_CIFAR10/
```

Pre-processing

```
print(f"Train size:, {X_train.shape[0]}, {X_train.shape[1]}")
```

```
## Train size:, 50000, 3072
```

```
print(f"Train size:, {X_test.shape[0]}, {X_test.shape[1]}")
```

```
## Train size:, 10000, 3072
```

VERY big dataset

Downsample the data so we can experiment more easily

Save the full dataset so we can train on the full training set later and do a blind test on the full test set.

```
X_train_full = X_train
y_train_full = y_train
X_test_full = X_test
y_test_full = y_test

subsample_rate = 0.02

X_train, _, y_train, _ = train_test_split(X_train, y_train, stratify=y_train, train_size=subsample_rate

X_test, _, y_test, _ = train_test_split(X_test, y_test, stratify=y_test, train_size=subsample_rate, ran

#We are using a subsample of the data
X_train.shape
```

```
## (1000, 3072)
```

```
knn_sk = KNeighborsClassifier(n_jobs=-1)

n_neighbors_range=list(range(1,6))
p_range=list(range(1,4))
parameters={'n_neighbors':n_neighbors_range, 'p':p_range}

np.random.seed(42) # for multiple runs of the same model training, you should re-execute this line
scores = ['accuracy']  # limited to accuracy for now but could use other metrics such as precision, rec
start_time = datetime.now()

for score in scores:
    print("# Tuning hyper-parameters for %s" % score)
    print()

    gridSearch = GridSearchCV(estimator=knn_sk,
                          param_grid=parameters,
                          cv=5,
                          #refit=False,
                          return_train_score=False,
                          verbose=1,
                          n_jobs=-1  #use multiple CPUs; divide and conquer!
                          )
    gridSearch.fit(X_train, y_train)
    print("Best parameters set found on development set:")
    print()
    print(gridSearch.best_params_)
    print()
    print("Grid scores on development set:")
    print()
    print(f'{score}  (+/-stdev*2)     hyper-params')

    means = gridSearch.cv_results_['mean_test_score']  #access elements of the grid search results dict
    stds = gridSearch.cv_results_['std_test_score']
    for mean, std, params in zip(means, stds, gridSearch.cv_results_['params']):
        print("%0.3f (+/-%0.03f) for %r"
              % (mean, std * 2, params))
    print()
    end_time = datetime.now()
    wallTimeInSecondsTrain = (end_time - start_time).total_seconds()

    print("Detailed classification report:")
    print()
    print("The model is trained on the full development set.")
    print("The scores are computed on the full evaluation set.")
    print()
    start_time = datetime.now()
    y_true, y_pred = y_test, gridSearch.predict(X_test)
    end_time = datetime.now()
    wallTimeInSecondsTest = (end_time - start_time).total_seconds()
    print(classification_report(y_true, y_pred))  # more detailed breakdown of the test perf; optional
    print()
    trainAcc = gridSearch.best_score_ #CV accuracy score for best hyperparameter combo
```

```
    testAcc  = accuracy_score(y_true, y_pred)
    experimentLog.loc[len(experimentLog)] =["knn", "Cifar10", f"{trainAcc*100:8.2f}%",
    f"{testAcc*100:8.2f}%",
    f"{wallTimeInSecondsTrain:8.2f} secs", f"{wallTimeInSecondsTest:8.2f} secs",
    f"{gridSearch.best_params_}","5-foldCV-based gridSearch BEST model"]
```

```
## # Tuning hyper-parameters for accuracy
##
## Fitting 5 folds for each of 15 candidates, totalling 75 fits
## GridSearchCV(cv=5, estimator=KNeighborsClassifier(n_jobs=-1), n_jobs=-1,
##              param_grid={'n_neighbors': [1, 2, 3, 4, 5], 'p': [1, 2, 3]},
##              verbose=1)
## Best parameters set found on development set:
##
## {'n_neighbors': 4, 'p': 1}
##
## Grid scores on development set:
##
## accuracy  (+/-stdev*2)     hyper-params
## 0.236 (+/-0.039) for {'n_neighbors': 1, 'p': 1}
## 0.236 (+/-0.019) for {'n_neighbors': 1, 'p': 2}
## 0.231 (+/-0.023) for {'n_neighbors': 1, 'p': 3}
## 0.211 (+/-0.027) for {'n_neighbors': 2, 'p': 1}
## 0.204 (+/-0.042) for {'n_neighbors': 2, 'p': 2}
## 0.186 (+/-0.038) for {'n_neighbors': 2, 'p': 3}
## 0.225 (+/-0.030) for {'n_neighbors': 3, 'p': 1}
## 0.237 (+/-0.042) for {'n_neighbors': 3, 'p': 2}
## 0.221 (+/-0.028) for {'n_neighbors': 3, 'p': 3}
## 0.257 (+/-0.043) for {'n_neighbors': 4, 'p': 1}
## 0.240 (+/-0.020) for {'n_neighbors': 4, 'p': 2}
## 0.227 (+/-0.045) for {'n_neighbors': 4, 'p': 3}
## 0.247 (+/-0.034) for {'n_neighbors': 5, 'p': 1}
## 0.254 (+/-0.026) for {'n_neighbors': 5, 'p': 2}
## 0.244 (+/-0.039) for {'n_neighbors': 5, 'p': 3}
##
## Detailed classification report:
##
## The model is trained on the full development set.
## The scores are computed on the full evaluation set.
##
##               precision    recall  f1-score   support
##
##            0       0.21      0.45      0.29        20
##            1       1.00      0.05      0.10        20
##            2       0.16      0.35      0.22        20
##            3       0.00      0.00      0.00        20
##            4       0.25      0.40      0.31        20
##            5       0.00      0.00      0.00        20
##            6       0.20      0.20      0.20        20
##            7       0.20      0.05      0.08        20
##            8       0.33      0.55      0.42        20
##            9       0.67      0.30      0.41        20
##
```

```
##     accuracy                           0.23       200
##    macro avg       0.30      0.23      0.20       200
## weighted avg       0.30      0.23      0.20       200
##
##
##
## [Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
## [Parallel(n_jobs=-1)]: Done  42 tasks      | elapsed:  1.6min
## [Parallel(n_jobs=-1)]: Done  75 out of  75 | elapsed:  3.0min finished
```

```
experiments= pd.DataFrame(experimentLog)
```

```
print(py$experiments)
```

```
##   Model Dataset  TrainAcc   TestAcc TrainTime(sec) TestTime(sec)
## 0   knn  Digits    98.59%    98.67%      7.24 secs     0.06 secs
## 1   knn Cifar10    25.70%    23.50%    183.22 secs     0.48 secs
##                        Param                          Description
## 0 {'n_neighbors': 3, 'p': 2} 5-foldCV-based gridSearch BEST model
## 1 {'n_neighbors': 4, 'p': 1} 5-foldCV-based gridSearch BEST model
```

```
best_p = gridSearch.best_params_["p"]
best_n = gridSearch.best_params_["n_neighbors"]
print("best_p: ",best_p)
```

```
## best_p:  1
```

```
print("best_n: ",best_n)
```

```
## best_n:  4
```

```
# have a look at CV dictionary of results
gridSearch.cv_results_
```

```
## {'mean_fit_time': array([0.40846229, 0.48116183, 0.5994421 , 0.54663181, 0.60157738,
##        0.6122716 , 0.62582102, 0.67835073, 0.62017665, 0.62344966,
##        0.69915233, 0.66422281, 0.72007842, 0.72842474, 0.65272498]), 'std_fit_time': array([0.161174
##        0.03217384, 0.07828906, 0.07429966, 0.04339658, 0.02925731,
##        0.03652525, 0.0732473 , 0.07295917, 0.01586388, 0.03136254]), 'mean_score_time': array([ 1.35
##        25.21221123,  1.122155  ,  1.12680888, 24.47440257,  1.01586494,
##         1.08730521, 24.48001781,  0.95362797,  1.03388863, 20.54953356]), 'std_score_time': array([0
##        0.68618616, 0.08450413, 0.04434587, 0.4962324 , 0.09706166,
##        0.05790411, 0.51057239, 0.07912226, 0.02769341, 5.86703771]), 'param_n_neighbors': masked_arra
##             mask=[False, False, False, False, False, False, False, False,
##                   False, False, False, False, False, False, False],
##        fill_value='?',
##            dtype=object), 'param_p': masked_array(data=[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3
##             mask=[False, False, False, False, False, False, False, False,
##                   False, False, False, False, False, False, False],
##        fill_value='?',
```

```
##             dtype=object), 'params': [{'n_neighbors': 1, 'p': 1}, {'n_neighbors': 1, 'p': 2}, {'n_nei
##        0.22 , 0.245, 0.22 , 0.245, 0.275, 0.265]), 'split1_test_score': array([0.225, 0.22 , 0.22 , 0
##        0.25 , 0.24 , 0.22 , 0.235, 0.235, 0.235]), 'split2_test_score': array([0.215, 0.245, 0.215, 0
##        0.285, 0.255, 0.27 , 0.28 , 0.26 , 0.27 ]), 'split3_test_score': array([0.27 , 0.24 , 0.24 , 0
##        0.265, 0.225, 0.205, 0.24 , 0.25 , 0.225]), 'split4_test_score': array([0.245, 0.245, 0.235, 0
##        0.265, 0.235, 0.22 , 0.235, 0.25 , 0.225]), 'mean_test_score': array([0.236, 0.236, 0.231, 0.1
##        0.257, 0.24 , 0.227, 0.247, 0.254, 0.244]), 'std_test_score': array([0.01959592, 0.00969536, 0
##        0.01881489, 0.01516575, 0.02111871, 0.01392839, 0.02158703,
##        0.01      , 0.02227106, 0.01691153, 0.01319091, 0.01959592]), 'rank_test_score': array([ 7,  1
##       dtype=int32)}
```

```
gridSearch.best_score_
```

```
## 0.257
```

```
acc = gridSearch.best_score_
print("KNN Grid Search Sklearn", np.round(acc, 3))
```

```
## KNN Grid Search Sklearn 0.257
```

Visualize the performance metrics along each hyperparameter

```
plt.figure(figsize=(8, 5))
```

```
## <Figure size 800x500 with 0 Axes>
```

```
sc = plt.scatter(gridSearch.cv_results_["param_n_neighbors"],
                 gridSearch.cv_results_["param_p"],
                 c=gridSearch.cv_results_["mean_test_score"],
                 cmap = plt.get_cmap("RdYlBu_r"),
                 s=1500)
plt.colorbar(sc)
```

```
## <matplotlib.colorbar.Colorbar object at 0x122fa5160>
```

```
plt.grid(True)
plt.gca().set_axisbelow(True)
plt.xlabel("Number of neighbors")
```

```
## Text(0.5, 0, 'Number of neighbors')
```

```
plt.ylabel("p")
```

```
## Text(0, 0.5, 'p')
```

```
plt.title("5-fold cross validation accuracy\n under different hyperparameter settings")
```

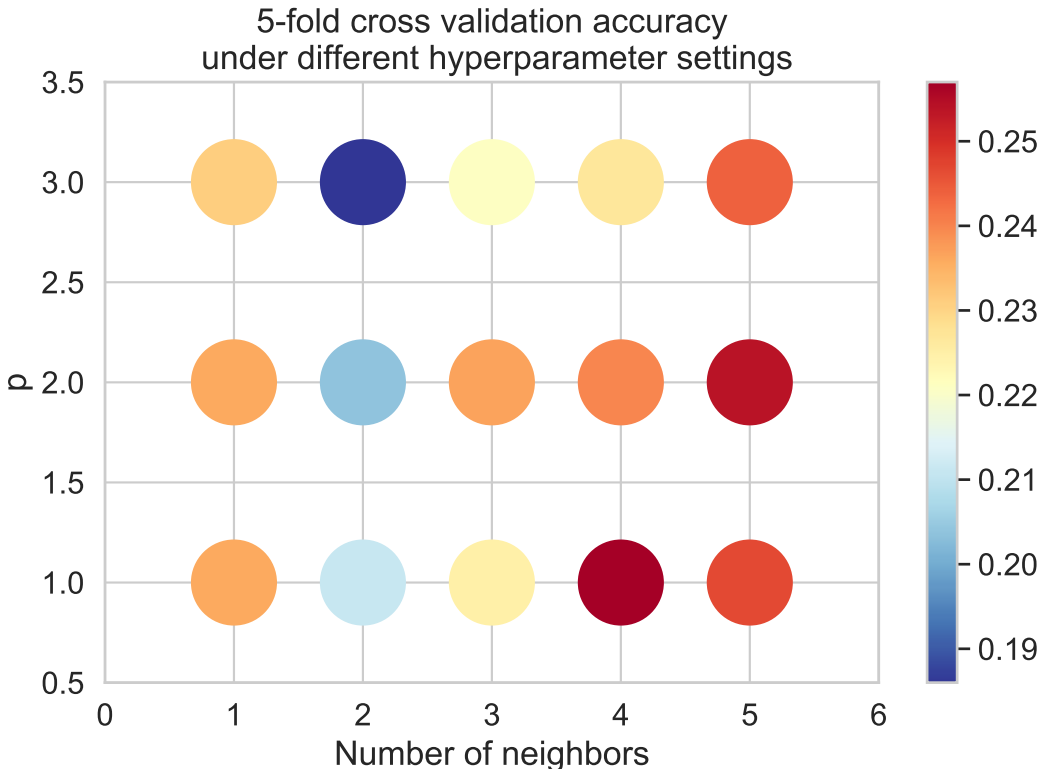
```
## Text(0.5, 1.0, '5-fold cross validation accuracy\n under different hyperparameter settings')
```

```
plt.xlim([0, 6])
```

```
## (0.0, 6.0)
```

```
plt.ylim([0.5, 3.5]);

plt.show()
```



Let's choose the best p^* and see how accuracy depends on the number of nearest neightbors for a KNN classifier p

```
p_idx = gridSearch.cv_results_["param_p"].data == best_p

plt.figure(figsize=(8,6))
```

```
## <Figure size 800x600 with 0 Axes>
```

```
plt.plot(gridSearch.cv_results_["param_n_neighbors"].data[p_idx],
         gridSearch.cv_results_["mean_test_score"][p_idx],
         "o-")
```

```
## [<matplotlib.lines.Line2D object at 0x124693340>]
```

```
plt.plot([best_n], [gridSearch.cv_results_["mean_test_score"][p_idx][best_n - 1]], "or", markersize=15,
```

```
## [<matplotlib.lines.Line2D object at 0x124693700>]
```

```
plt.xlabel("Number of neighbors")
```

```
## Text(0.5, 0, 'Number of neighbors')
```

```
plt.ylabel("Accuracy")
```

```
## Text(0, 0.5, 'Accuracy')
```

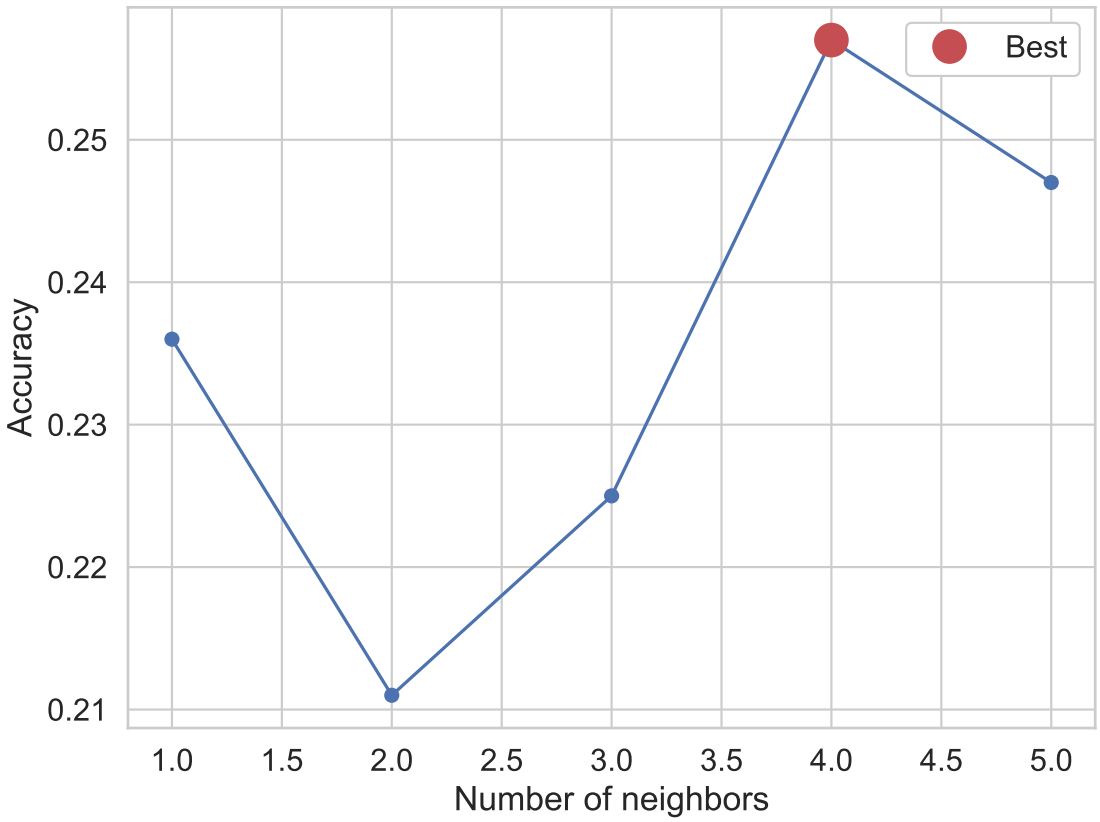
```
plt.title("")
```

```
## Text(0.5, 1.0, '')
```

```
plt.grid(True)
plt.legend(numpoints=1)
```

```
## <matplotlib.legend.Legend object at 0x122f247c0>
```

```
plt.show()
```



Refit estimator with best parameters using 100% of the training data

Note: This is required because we ran GridSearchCV with 2% of the available training data.

In practice make sure X_train, y_train has 100% of the training data. Note this is not the case here.

```
knn= KNeighborsClassifier(n_neighbors=gridSearch.best_params_['n_neighbors'],
p=gridSearch.best_params_['p'], n_jobs=-1)

start_time=datetime.now()
knn.fit(X_train_full,y_train_full)
```

```
## KNeighborsClassifier(n_jobs=-1, n_neighbors=4, p=1)
```

```
end_time=datetime.now()
wallTimeInSecondsTrain = (end_time - start_time).total_seconds()
print("Training time (s): ", wallTimeInSecondsTrain)
```

```
## Training time (s):  44.89056
```

```
start_time = datetime.now()
y_preds_full = knn.predict(X_test_full)
end_time = datetime.now()
wallTimeInSecondsTest = (end_time - start_time).total_seconds()
print("Test data prediction time (s): ", wallTimeInSecondsTest)
```

```
## Test data prediction time (s):  1382.480316
```

```
testAcc = accuracy_score(y_test_full, y_preds_full)
```

```
experimentLog.loc[len(experimentLog)] =["knn", "Cifar-10-FullData", f"{trainAcc*100:8.2f}%", f"{testAcc
                                        f"{wallTimeInSecondsTrain:8.2f} secs", f"{wallTimeInSecondsTest:8.2
                                        f"{gridSearch.best_params_}","Best model trained on 100% of trainin

experiments=pd.DataFrame(experimentLog)
```

```
print(py$experiments)
```

```
##   Model           Dataset  TrainAcc   TestAcc TrainTime(sec) TestTime(sec)
## 0   knn            Digits    98.59%    98.67%      7.24 secs     0.06 secs
## 1   knn           Cifar10    25.70%    23.50%    183.22 secs     0.48 secs
## 2   knn Cifar-10-FullData    25.70%    36.84%     44.89 secs  1382.48 secs
##                       Param                                  Description
## 0 {'n_neighbors': 3, 'p': 2}        5-foldCV-based gridSearch BEST model
## 1 {'n_neighbors': 4, 'p': 1}        5-foldCV-based gridSearch BEST model
## 2 {'n_neighbors': 4, 'p': 1} Best model trained on 100% of training data
```

Let's try our luck with Weighted KNN and see if accuracy on the test set can be improved

```
np.random.seed(42)

knn_sk = KNeighborsClassifier(n_jobs=-1)
weight_options = ['uniform', 'distance']
n_neighbors_range = list(range(1,6))
p_range = list (range(1,4))

#parameters = {'n_neighbors': n_neighbors_range, 'p': p_range, 'weights'=weight_options}
param_grid = dict(n_neighbors=n_neighbors_range,
weights=weight_options,p=p_range)


start_time = datetime.now()
knn_weighted_gs = GridSearchCV(knn_sk, param_grid, cv=5, verbose=2)
knn_weighted_gs.fit(X_train, y_train)
```

```
## Fitting 5 folds for each of 30 candidates, totalling 150 fits
## [CV] n_neighbors=1, p=1, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=1, weights=uniform, total=   4.6s
## [CV] n_neighbors=1, p=1, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=1, p=1, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=1, p=1, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=1, p=1, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=1, weights=uniform, total=   0.7s
## [CV] n_neighbors=1, p=1, weights=distance .............................
## [CV] .............. n_neighbors=1, p=1, weights=distance, total=   0.7s
## [CV] n_neighbors=1, p=1, weights=distance .............................
## [CV] .............. n_neighbors=1, p=1, weights=distance, total=   0.8s
## [CV] n_neighbors=1, p=1, weights=distance .............................
## [CV] .............. n_neighbors=1, p=1, weights=distance, total=   0.8s
## [CV] n_neighbors=1, p=1, weights=distance .............................
## [CV] .............. n_neighbors=1, p=1, weights=distance, total=   0.7s
## [CV] n_neighbors=1, p=1, weights=distance .............................
## [CV] .............. n_neighbors=1, p=1, weights=distance, total=   0.7s
## [CV] n_neighbors=1, p=2, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=2, weights=uniform, total=   0.7s
## [CV] n_neighbors=1, p=2, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=2, weights=uniform, total=   0.8s
## [CV] n_neighbors=1, p=2, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=2, weights=uniform, total=   0.8s
## [CV] n_neighbors=1, p=2, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=2, weights=uniform, total=   0.7s
## [CV] n_neighbors=1, p=2, weights=uniform ..............................
## [CV] .............. n_neighbors=1, p=2, weights=uniform, total=   0.8s
## [CV] n_neighbors=1, p=2, weights=distance .............................
## [CV] .............. n_neighbors=1, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=1, p=2, weights=distance .............................
## [CV] .............. n_neighbors=1, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=1, p=2, weights=distance .............................
## [CV] .............. n_neighbors=1, p=2, weights=distance, total=   0.8s
```

```
## [CV] n_neighbors=1, p=2, weights=distance .............................
## [CV] ............. n_neighbors=1, p=2, weights=distance, total=   0.7s
## [CV] n_neighbors=1, p=2, weights=distance .............................
## [CV] ............. n_neighbors=1, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=1, p=3, weights=uniform ..............................
## [CV] ............... n_neighbors=1, p=3, weights=uniform, total=   8.8s
## [CV] n_neighbors=1, p=3, weights=uniform ..............................
## [CV] ............... n_neighbors=1, p=3, weights=uniform, total=   7.6s
## [CV] n_neighbors=1, p=3, weights=uniform ..............................
## [CV] ............... n_neighbors=1, p=3, weights=uniform, total=   7.0s
## [CV] n_neighbors=1, p=3, weights=uniform ..............................
## [CV] ............... n_neighbors=1, p=3, weights=uniform, total=   7.8s
## [CV] n_neighbors=1, p=3, weights=uniform ..............................
## [CV] ............... n_neighbors=1, p=3, weights=uniform, total=   8.8s
## [CV] n_neighbors=1, p=3, weights=distance ..............................
## [CV] ............. n_neighbors=1, p=3, weights=distance, total=   7.7s
## [CV] n_neighbors=1, p=3, weights=distance ..............................
## [CV] ............. n_neighbors=1, p=3, weights=distance, total=   9.2s
## [CV] n_neighbors=1, p=3, weights=distance ..............................
## [CV] ............. n_neighbors=1, p=3, weights=distance, total=   8.1s
## [CV] n_neighbors=1, p=3, weights=distance ..............................
## [CV] ............. n_neighbors=1, p=3, weights=distance, total=   6.7s
## [CV] n_neighbors=1, p=3, weights=distance ..............................
## [CV] ............. n_neighbors=1, p=3, weights=distance, total=   7.4s
## [CV] n_neighbors=2, p=1, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=1, weights=uniform, total=   1.0s
## [CV] n_neighbors=2, p=1, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=1, weights=uniform, total=   1.5s
## [CV] n_neighbors=2, p=1, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=1, weights=uniform, total=   1.4s
## [CV] n_neighbors=2, p=1, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=1, weights=uniform, total=   0.9s
## [CV] n_neighbors=2, p=1, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=1, weights=uniform, total=   1.1s
## [CV] n_neighbors=2, p=1, weights=distance ..............................
## [CV] ............. n_neighbors=2, p=1, weights=distance, total=   0.8s
## [CV] n_neighbors=2, p=1, weights=distance ..............................
## [CV] ............. n_neighbors=2, p=1, weights=distance, total=   0.8s
## [CV] n_neighbors=2, p=1, weights=distance ..............................
## [CV] ............. n_neighbors=2, p=1, weights=distance, total=   0.8s
## [CV] n_neighbors=2, p=1, weights=distance ..............................
## [CV] ............. n_neighbors=2, p=1, weights=distance, total=   0.8s
## [CV] n_neighbors=2, p=1, weights=distance ..............................
## [CV] ............. n_neighbors=2, p=1, weights=distance, total=   1.0s
## [CV] n_neighbors=2, p=2, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=2, weights=uniform, total=   0.8s
## [CV] n_neighbors=2, p=2, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=2, weights=uniform, total=   0.7s
## [CV] n_neighbors=2, p=2, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=2, weights=uniform, total=   0.8s
## [CV] n_neighbors=2, p=2, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=2, weights=uniform, total=   0.7s
## [CV] n_neighbors=2, p=2, weights=uniform ..............................
## [CV] ............... n_neighbors=2, p=2, weights=uniform, total=   0.7s
```

```
## [CV] n_neighbors=2, p=2, weights=distance .............................
## [CV] ............. n_neighbors=2, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=2, p=2, weights=distance .............................
## [CV] ............. n_neighbors=2, p=2, weights=distance, total=   0.7s
## [CV] n_neighbors=2, p=2, weights=distance .............................
## [CV] ............. n_neighbors=2, p=2, weights=distance, total=   0.7s
## [CV] n_neighbors=2, p=2, weights=distance .............................
## [CV] ............. n_neighbors=2, p=2, weights=distance, total=   0.7s
## [CV] n_neighbors=2, p=2, weights=distance .............................
## [CV] ............. n_neighbors=2, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=2, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=2, p=3, weights=uniform, total=  10.5s
## [CV] n_neighbors=2, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=2, p=3, weights=uniform, total=   7.4s
## [CV] n_neighbors=2, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=2, p=3, weights=uniform, total=   7.8s
## [CV] n_neighbors=2, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=2, p=3, weights=uniform, total=   7.2s
## [CV] n_neighbors=2, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=2, p=3, weights=uniform, total=  12.2s
## [CV] n_neighbors=2, p=3, weights=distance .............................
## [CV] ............. n_neighbors=2, p=3, weights=distance, total=   8.2s
## [CV] n_neighbors=2, p=3, weights=distance .............................
## [CV] ............. n_neighbors=2, p=3, weights=distance, total=   6.9s
## [CV] n_neighbors=2, p=3, weights=distance .............................
## [CV] ............. n_neighbors=2, p=3, weights=distance, total=   6.7s
## [CV] n_neighbors=2, p=3, weights=distance .............................
## [CV] ............. n_neighbors=2, p=3, weights=distance, total=   9.4s
## [CV] n_neighbors=2, p=3, weights=distance .............................
## [CV] ............. n_neighbors=2, p=3, weights=distance, total=   6.9s
## [CV] n_neighbors=3, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=3, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=3, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=3, p=1, weights=uniform, total=   1.1s
## [CV] n_neighbors=3, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=3, p=1, weights=uniform, total=   0.9s
## [CV] n_neighbors=3, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=3, p=1, weights=uniform, total=   1.6s
## [CV] n_neighbors=3, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=3, p=1, weights=uniform, total=   1.2s
## [CV] n_neighbors=3, p=1, weights=distance .............................
## [CV] ............. n_neighbors=3, p=1, weights=distance, total=   1.3s
## [CV] n_neighbors=3, p=1, weights=distance .............................
## [CV] ............. n_neighbors=3, p=1, weights=distance, total=   1.2s
## [CV] n_neighbors=3, p=1, weights=distance .............................
## [CV] ............. n_neighbors=3, p=1, weights=distance, total=   1.5s
## [CV] n_neighbors=3, p=1, weights=distance .............................
## [CV] ............. n_neighbors=3, p=1, weights=distance, total=   1.1s
## [CV] n_neighbors=3, p=1, weights=distance .............................
## [CV] ............. n_neighbors=3, p=1, weights=distance, total=   1.1s
## [CV] n_neighbors=3, p=2, weights=uniform ..............................
## [CV] ............. n_neighbors=3, p=2, weights=uniform, total=   1.2s
## [CV] n_neighbors=3, p=2, weights=uniform ..............................
## [CV] ............. n_neighbors=3, p=2, weights=uniform, total=   1.5s
```

```
## [CV] n_neighbors=3, p=2, weights=uniform .............................
## [CV] .............. n_neighbors=3, p=2, weights=uniform, total=   1.1s
## [CV] n_neighbors=3, p=2, weights=uniform .............................
## [CV] .............. n_neighbors=3, p=2, weights=uniform, total=   1.1s
## [CV] n_neighbors=3, p=2, weights=uniform .............................
## [CV] .............. n_neighbors=3, p=2, weights=uniform, total=   1.1s
## [CV] n_neighbors=3, p=2, weights=distance ............................
## [CV] ............. n_neighbors=3, p=2, weights=distance, total=   1.0s
## [CV] n_neighbors=3, p=2, weights=distance ............................
## [CV] ............. n_neighbors=3, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=3, p=2, weights=distance ............................
## [CV] ............. n_neighbors=3, p=2, weights=distance, total=   1.1s
## [CV] n_neighbors=3, p=2, weights=distance ............................
## [CV] ............. n_neighbors=3, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=3, p=2, weights=distance ............................
## [CV] ............. n_neighbors=3, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=3, p=3, weights=uniform .............................
## [CV] .............. n_neighbors=3, p=3, weights=uniform, total=   8.5s
## [CV] n_neighbors=3, p=3, weights=uniform .............................
## [CV] .............. n_neighbors=3, p=3, weights=uniform, total=   6.8s
## [CV] n_neighbors=3, p=3, weights=uniform .............................
## [CV] .............. n_neighbors=3, p=3, weights=uniform, total=   7.0s
## [CV] n_neighbors=3, p=3, weights=uniform .............................
## [CV] .............. n_neighbors=3, p=3, weights=uniform, total=   7.3s
## [CV] n_neighbors=3, p=3, weights=uniform .............................
## [CV] .............. n_neighbors=3, p=3, weights=uniform, total=   9.9s
## [CV] n_neighbors=3, p=3, weights=distance ............................
## [CV] ............. n_neighbors=3, p=3, weights=distance, total=   7.4s
## [CV] n_neighbors=3, p=3, weights=distance ............................
## [CV] ............. n_neighbors=3, p=3, weights=distance, total=   7.4s
## [CV] n_neighbors=3, p=3, weights=distance ............................
## [CV] ............. n_neighbors=3, p=3, weights=distance, total=  10.2s
## [CV] n_neighbors=3, p=3, weights=distance ............................
## [CV] ............. n_neighbors=3, p=3, weights=distance, total=   7.4s
## [CV] n_neighbors=3, p=3, weights=distance ............................
## [CV] ............. n_neighbors=3, p=3, weights=distance, total=   9.8s
## [CV] n_neighbors=4, p=1, weights=uniform .............................
## [CV] .............. n_neighbors=4, p=1, weights=uniform, total=   0.7s
## [CV] n_neighbors=4, p=1, weights=uniform .............................
## [CV] .............. n_neighbors=4, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=4, p=1, weights=uniform .............................
## [CV] .............. n_neighbors=4, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=4, p=1, weights=uniform .............................
## [CV] .............. n_neighbors=4, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=4, p=1, weights=uniform .............................
## [CV] .............. n_neighbors=4, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=4, p=1, weights=distance ............................
## [CV] ............. n_neighbors=4, p=1, weights=distance, total=   0.9s
## [CV] n_neighbors=4, p=1, weights=distance ............................
## [CV] ............. n_neighbors=4, p=1, weights=distance, total=   1.3s
## [CV] n_neighbors=4, p=1, weights=distance ............................
## [CV] ............. n_neighbors=4, p=1, weights=distance, total=   1.0s
## [CV] n_neighbors=4, p=1, weights=distance ............................
## [CV] ............. n_neighbors=4, p=1, weights=distance, total=   1.0s
```

```
## [CV] n_neighbors=4, p=1, weights=distance .............................
## [CV] ............. n_neighbors=4, p=1, weights=distance, total=   1.0s
## [CV] n_neighbors=4, p=2, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=2, weights=uniform, total=   1.0s
## [CV] n_neighbors=4, p=2, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=2, weights=uniform, total=   0.9s
## [CV] n_neighbors=4, p=2, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=2, weights=uniform, total=   1.0s
## [CV] n_neighbors=4, p=2, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=2, weights=uniform, total=   0.9s
## [CV] n_neighbors=4, p=2, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=2, weights=uniform, total=   1.0s
## [CV] n_neighbors=4, p=2, weights=distance .............................
## [CV] ............. n_neighbors=4, p=2, weights=distance, total=   1.0s
## [CV] n_neighbors=4, p=2, weights=distance .............................
## [CV] ............. n_neighbors=4, p=2, weights=distance, total=   1.6s
## [CV] n_neighbors=4, p=2, weights=distance .............................
## [CV] ............. n_neighbors=4, p=2, weights=distance, total=   1.5s
## [CV] n_neighbors=4, p=2, weights=distance .............................
## [CV] ............. n_neighbors=4, p=2, weights=distance, total=   1.0s
## [CV] n_neighbors=4, p=2, weights=distance .............................
## [CV] ............. n_neighbors=4, p=2, weights=distance, total=   1.0s
## [CV] n_neighbors=4, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=3, weights=uniform, total=   8.5s
## [CV] n_neighbors=4, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=3, weights=uniform, total=   8.5s
## [CV] n_neighbors=4, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=3, weights=uniform, total=  11.1s
## [CV] n_neighbors=4, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=3, weights=uniform, total=  12.2s
## [CV] n_neighbors=4, p=3, weights=uniform ..............................
## [CV] ............. n_neighbors=4, p=3, weights=uniform, total=   7.1s
## [CV] n_neighbors=4, p=3, weights=distance .............................
## [CV] ............. n_neighbors=4, p=3, weights=distance, total=   7.7s
## [CV] n_neighbors=4, p=3, weights=distance .............................
## [CV] ............. n_neighbors=4, p=3, weights=distance, total=   7.6s
## [CV] n_neighbors=4, p=3, weights=distance .............................
## [CV] ............. n_neighbors=4, p=3, weights=distance, total=   7.2s
## [CV] n_neighbors=4, p=3, weights=distance .............................
## [CV] ............. n_neighbors=4, p=3, weights=distance, total=   7.4s
## [CV] n_neighbors=4, p=3, weights=distance .............................
## [CV] ............. n_neighbors=4, p=3, weights=distance, total=   8.6s
## [CV] n_neighbors=5, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=5, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=5, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=5, p=1, weights=uniform, total=   0.9s
## [CV] n_neighbors=5, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=5, p=1, weights=uniform, total=   1.1s
## [CV] n_neighbors=5, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=5, p=1, weights=uniform, total=   0.8s
## [CV] n_neighbors=5, p=1, weights=uniform ..............................
## [CV] ............. n_neighbors=5, p=1, weights=uniform, total=   1.0s
## [CV] n_neighbors=5, p=1, weights=distance .............................
## [CV] ............. n_neighbors=5, p=1, weights=distance, total=   0.8s
```

```
## [CV] n_neighbors=5, p=1, weights=distance ............................
## [CV] ............. n_neighbors=5, p=1, weights=distance, total=   0.9s
## [CV] n_neighbors=5, p=1, weights=distance ............................
## [CV] ............. n_neighbors=5, p=1, weights=distance, total=   0.8s
## [CV] n_neighbors=5, p=1, weights=distance ............................
## [CV] ............. n_neighbors=5, p=1, weights=distance, total=   0.8s
## [CV] n_neighbors=5, p=1, weights=distance ............................
## [CV] ............. n_neighbors=5, p=1, weights=distance, total=   0.7s
## [CV] n_neighbors=5, p=2, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=2, weights=uniform, total=   0.9s
## [CV] n_neighbors=5, p=2, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=2, weights=uniform, total=   0.9s
## [CV] n_neighbors=5, p=2, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=2, weights=uniform, total=   0.7s
## [CV] n_neighbors=5, p=2, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=2, weights=uniform, total=   0.8s
## [CV] n_neighbors=5, p=2, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=2, weights=uniform, total=   0.7s
## [CV] n_neighbors=5, p=2, weights=distance ............................
## [CV] ............. n_neighbors=5, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=5, p=2, weights=distance ............................
## [CV] ............. n_neighbors=5, p=2, weights=distance, total=   0.7s
## [CV] n_neighbors=5, p=2, weights=distance ............................
## [CV] ............. n_neighbors=5, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=5, p=2, weights=distance ............................
## [CV] ............. n_neighbors=5, p=2, weights=distance, total=   0.8s
## [CV] n_neighbors=5, p=2, weights=distance ............................
## [CV] ............. n_neighbors=5, p=2, weights=distance, total=   0.7s
## [CV] n_neighbors=5, p=3, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=3, weights=uniform, total=   7.0s
## [CV] n_neighbors=5, p=3, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=3, weights=uniform, total=   6.8s
## [CV] n_neighbors=5, p=3, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=3, weights=uniform, total=   6.9s
## [CV] n_neighbors=5, p=3, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=3, weights=uniform, total=   7.8s
## [CV] n_neighbors=5, p=3, weights=uniform .............................
## [CV] ............. n_neighbors=5, p=3, weights=uniform, total=   7.6s
## [CV] n_neighbors=5, p=3, weights=distance ............................
## [CV] ............. n_neighbors=5, p=3, weights=distance, total=   6.9s
## [CV] n_neighbors=5, p=3, weights=distance ............................
## [CV] ............. n_neighbors=5, p=3, weights=distance, total=   7.0s
## [CV] n_neighbors=5, p=3, weights=distance ............................
## [CV] ............. n_neighbors=5, p=3, weights=distance, total=   7.4s
## [CV] n_neighbors=5, p=3, weights=distance ............................
## [CV] ............. n_neighbors=5, p=3, weights=distance, total=   6.9s
## [CV] n_neighbors=5, p=3, weights=distance ............................
## [CV] ............. n_neighbors=5, p=3, weights=distance, total=   6.7s
## GridSearchCV(cv=5, estimator=KNeighborsClassifier(n_jobs=-1),
##              param_grid={'n_neighbors': [1, 2, 3, 4, 5], 'p': [1, 2, 3],
##                          'weights': ['uniform', 'distance']},
##              verbose=2)
##
## [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
## [Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    4.6s remaining:    0.0s
## [Parallel(n_jobs=1)]: Done 150 out of 150 | elapsed:  8.3min finished
```

```
end_time = datetime.now()
wallTimeInSecondsTrain = (end_time - start_time).total_seconds()
```

```
print('Best parameters:', knn_weighted_gs.best_params_)
```

```
## Best parameters: {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
```

```
train_acc = knn_weighted_gs.best_score_

start_time = datetime.now()
test_preds = knn_weighted_gs.best_estimator_.predict(X_test)
test_acc = accuracy_score(y_test, test_preds)
end_time = datetime.now()
wallTimeInSecondsTest = (end_time - start_time).total_seconds()
```

```
experimentLog.loc[len(experimentLog)] =["KNN Weighted Test GridSearch", "Cifar10", f"{train_acc*100:8.2
                                        f"{wallTimeInSecondsTrain:8.2f} secs", f"{wallTimeInSecondsTest:8.2
                                        f"{knn_weighted_gs.best_params_}","5-foldCV-based gridSearch BEST m
```

```
experiments=pd.DataFrame(experimentLog)
```

```
print(py$experiments)
```

```
##                          Model           Dataset  TrainAcc   TestAcc
## 0                          knn            Digits    98.59%    98.67%
## 1                          knn           Cifar10    25.70%    23.50%
## 2                          knn Cifar-10-FullData    25.70%    36.84%
## 3 KNN Weighted Test GridSearch           Cifar10    27.20%    26.50%
##   TrainTime(sec) TestTime(sec)
## 0      7.24 secs     0.06 secs
## 1    183.22 secs     0.48 secs
## 2     44.89 secs  1382.48 secs
## 3    496.79 secs     0.53 secs
##                                          Param
## 0                        {'n_neighbors': 3, 'p': 2}
## 1                        {'n_neighbors': 4, 'p': 1}
## 2                        {'n_neighbors': 4, 'p': 1}
## 3 {'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
##                                   Description
## 0        5-foldCV-based gridSearch BEST model
## 1        5-foldCV-based gridSearch BEST model
## 2 Best model trained on 100% of training data
## 3        5-foldCV-based gridSearch BEST model
```

The accuracy has improved for the 2% data. But not by much. SO we won't try weighted knn on full data. If you do want to try, the code is given below

```
#knn= KNeighborsClassifier(n_neighbors=knn_weighted_gs.best_params_['n_neighbors'],
#p=knn_weighted_gs.best_params_['p'], n_jobs=-1)
#knn= KNeighborsClassifier(n_neighbors=knn_weighted_gs.best_params_['n_neighbors'],
#p=knn_weighted_gs.best_params_['p'],
#weights=knn_weighted_gs.best_params_['weights'],
#n_jobs=-1)

#start_time=datetime.now()
#knn.fit(X_train_full,y_train_full)
#end_time=datetime.now()
#wallTimeInSecondsTrain = (end_time - start_time).total_seconds()
#print("Training time (s): ", wallTimeInSecondsTrain)

#start_time = datetime.now()
#y_preds_full = knn.predict(X_test_full)
#end_time = datetime.now()
#wallTimeInSecondsTest = (end_time - start_time).total_seconds()
#print("Test data prediction time (s): ", wallTimeInSecondsTest)
#testAcc = accuracy_score(y_test_full, y_preds_full)
```

KNN Regression using Boston Housing Data

```
from sklearn.datasets import load_boston
boston = load_boston()
print("Data shape: {}".format(boston.data.shape))
```

```
## Data shape: (506, 13)
```

```
X = pd.DataFrame(boston.data, columns=boston.feature_names)
y = boston.target

X.head()
```

```
##       CRIM    ZN  INDUS  CHAS    NOX  ...  RAD    TAX  PTRATIO       B  LSTAT
## 0  0.00632  18.0   2.31   0.0  0.538  ...  1.0  296.0     15.3  396.90   4.98
## 1  0.02731   0.0   7.07   0.0  0.469  ...  2.0  242.0     17.8  396.90   9.14
## 2  0.02729   0.0   7.07   0.0  0.469  ...  2.0  242.0     17.8  392.83   4.03
## 3  0.03237   0.0   2.18   0.0  0.458  ...  3.0  222.0     18.7  394.63   2.94
## 4  0.06905   0.0   2.18   0.0  0.458  ...  3.0  222.0     18.7  396.90   5.33
##
## [5 rows x 13 columns]
```

```
X.describe()
```

```
##              CRIM          ZN       INDUS  ...     PTRATIO           B       LSTAT
## count  506.000000  506.000000  506.000000  ...  506.000000  506.000000  506.000000
## mean     3.613524   11.363636   11.136779  ...   18.455534  356.674032   12.653063
## std      8.601545   23.322453    6.860353  ...    2.164946   91.294864    7.141062
## min      0.006320    0.000000    0.460000  ...   12.600000    0.320000    1.730000
## 25%      0.082045    0.000000    5.190000  ...   17.400000  375.377500    6.950000
## 50%      0.256510    0.000000    9.690000  ...   19.050000  391.440000   11.360000
```

```
## 75%      3.677083   12.500000   18.100000  ...   20.200000  396.225000   16.955000
## max     88.976200  100.000000   27.740000  ...   22.000000  396.900000   37.970000
##
## [8 rows x 13 columns]
```

```
from sklearn.preprocessing import MinMaxScaler
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=42)
scaler = MinMaxScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

np.random.seed(42)
scores = ['neg_mean_squared_error']
knn_reg=KNeighborsRegressor(n_jobs=-1)

weight_options = ['uniform', 'distance']
n_neighbors_range = list(range(1,11))
p_range = list (range(1,4))

parameters = {'knn__n_neighbors': n_neighbors_range,
              'knn__p':p_range ,
              'knn__weights': weight_options}

pipe = Pipeline([('standardize', StandardScaler()),('knn', knn_reg)])

knn_reg_gs = GridSearchCV(pipe, parameters, cv=3, refit=True, verbose=3)
```

```
knn_reg_gs.fit(X_train, y_train)
```

```
## Fitting 3 folds for each of 60 candidates, totalling 180 fits
## [CV] knn__n_neighbors=1, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=1, knn__p=1, knn__weights=uniform, score=0.684, total=   0.1s
## [CV] knn__n_neighbors=1, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=1, knn__p=1, knn__weights=uniform, score=0.702, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=1, knn__p=1, knn__weights=uniform, score=0.736, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=1, knn__p=1, knn__weights=distance, score=0.684, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=1, knn__p=1, knn__weights=distance, score=0.702, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=1, knn__p=1, knn__weights=distance, score=0.736, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=1, knn__p=2, knn__weights=uniform, score=0.744, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=1, knn__p=2, knn__weights=uniform, score=0.714, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=2, knn__weights=uniform ..............
```

```
## [CV]  knn__n_neighbors=1, knn__p=2, knn__weights=uniform, score=0.715, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=1, knn__p=2, knn__weights=distance, score=0.744, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=1, knn__p=2, knn__weights=distance, score=0.714, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=1, knn__p=2, knn__weights=distance, score=0.715, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=1, knn__p=3, knn__weights=uniform, score=0.748, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=1, knn__p=3, knn__weights=uniform, score=0.727, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=1, knn__p=3, knn__weights=uniform, score=0.693, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=1, knn__p=3, knn__weights=distance, score=0.748, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=1, knn__p=3, knn__weights=distance, score=0.727, total=   0.0s
## [CV] knn__n_neighbors=1, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=1, knn__p=3, knn__weights=distance, score=0.693, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=2, knn__p=1, knn__weights=uniform, score=0.772, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=2, knn__p=1, knn__weights=uniform, score=0.784, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=2, knn__p=1, knn__weights=uniform, score=0.756, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=2, knn__p=1, knn__weights=distance, score=0.784, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=2, knn__p=1, knn__weights=distance, score=0.792, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=2, knn__p=1, knn__weights=distance, score=0.791, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=2, knn__p=2, knn__weights=uniform, score=0.784, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=2, knn__p=2, knn__weights=uniform, score=0.814, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=2, knn__p=2, knn__weights=uniform, score=0.713, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=2, knn__p=2, knn__weights=distance, score=0.799, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=2, knn__p=2, knn__weights=distance, score=0.816, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=2, knn__p=2, knn__weights=distance, score=0.759, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=2, knn__p=3, knn__weights=uniform, score=0.779, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=2, knn__p=3, knn__weights=uniform, score=0.762, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=2, knn__p=3, knn__weights=uniform, score=0.658, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=2, knn__p=3, knn__weights=distance, score=0.788, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=2, knn__p=3, knn__weights=distance, score=0.762, total=   0.0s
## [CV] knn__n_neighbors=2, knn__p=3, knn__weights=distance .............
```

```
## [CV]  knn__n_neighbors=2, knn__p=3, knn__weights=distance, score=0.695, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=3, knn__p=1, knn__weights=uniform, score=0.740, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=3, knn__p=1, knn__weights=uniform, score=0.781, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=3, knn__p=1, knn__weights=uniform, score=0.762, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=3, knn__p=1, knn__weights=distance, score=0.768, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=3, knn__p=1, knn__weights=distance, score=0.795, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=3, knn__p=1, knn__weights=distance, score=0.810, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=3, knn__p=2, knn__weights=uniform, score=0.730, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=3, knn__p=2, knn__weights=uniform, score=0.790, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=3, knn__p=2, knn__weights=uniform, score=0.705, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=3, knn__p=2, knn__weights=distance, score=0.775, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=3, knn__p=2, knn__weights=distance, score=0.802, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=3, knn__p=2, knn__weights=distance, score=0.759, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=3, knn__p=3, knn__weights=uniform, score=0.744, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=3, knn__p=3, knn__weights=uniform, score=0.758, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=3, knn__p=3, knn__weights=uniform, score=0.708, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=3, knn__p=3, knn__weights=distance, score=0.779, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=3, knn__p=3, knn__weights=distance, score=0.769, total=   0.0s
## [CV] knn__n_neighbors=3, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=3, knn__p=3, knn__weights=distance, score=0.751, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=4, knn__p=1, knn__weights=uniform, score=0.726, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=4, knn__p=1, knn__weights=uniform, score=0.772, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=4, knn__p=1, knn__weights=uniform, score=0.667, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=4, knn__p=1, knn__weights=distance, score=0.767, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=4, knn__p=1, knn__weights=distance, score=0.804, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=4, knn__p=1, knn__weights=distance, score=0.756, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=4, knn__p=2, knn__weights=uniform, score=0.696, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=4, knn__p=2, knn__weights=uniform, score=0.763, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=2, knn__weights=uniform ..............
```

```
## [CV]  knn__n_neighbors=4, knn__p=2, knn__weights=uniform, score=0.653, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=4, knn__p=2, knn__weights=distance, score=0.765, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=4, knn__p=2, knn__weights=distance, score=0.788, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=4, knn__p=2, knn__weights=distance, score=0.728, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=4, knn__p=3, knn__weights=uniform, score=0.710, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=4, knn__p=3, knn__weights=uniform, score=0.768, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=4, knn__p=3, knn__weights=uniform, score=0.655, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=4, knn__p=3, knn__weights=distance, score=0.763, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=4, knn__p=3, knn__weights=distance, score=0.788, total=   0.0s
## [CV] knn__n_neighbors=4, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=4, knn__p=3, knn__weights=distance, score=0.716, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=5, knn__p=1, knn__weights=uniform, score=0.696, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=5, knn__p=1, knn__weights=uniform, score=0.769, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=5, knn__p=1, knn__weights=uniform, score=0.658, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=5, knn__p=1, knn__weights=distance, score=0.748, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=5, knn__p=1, knn__weights=distance, score=0.803, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=5, knn__p=1, knn__weights=distance, score=0.741, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=5, knn__p=2, knn__weights=uniform, score=0.705, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=5, knn__p=2, knn__weights=uniform, score=0.742, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=5, knn__p=2, knn__weights=uniform, score=0.636, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=5, knn__p=2, knn__weights=distance, score=0.763, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=5, knn__p=2, knn__weights=distance, score=0.783, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=5, knn__p=2, knn__weights=distance, score=0.725, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=5, knn__p=3, knn__weights=uniform, score=0.720, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=5, knn__p=3, knn__weights=uniform, score=0.744, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=5, knn__p=3, knn__weights=uniform, score=0.664, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=5, knn__p=3, knn__weights=distance, score=0.768, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=5, knn__p=3, knn__weights=distance, score=0.778, total=   0.0s
## [CV] knn__n_neighbors=5, knn__p=3, knn__weights=distance .............
```

```
## [CV]  knn__n_neighbors=5, knn__p=3, knn__weights=distance, score=0.725, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=6, knn__p=1, knn__weights=uniform, score=0.696, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=6, knn__p=1, knn__weights=uniform, score=0.757, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=6, knn__p=1, knn__weights=uniform, score=0.634, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=6, knn__p=1, knn__weights=distance, score=0.749, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=6, knn__p=1, knn__weights=distance, score=0.798, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=6, knn__p=1, knn__weights=distance, score=0.725, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=6, knn__p=2, knn__weights=uniform, score=0.713, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=6, knn__p=2, knn__weights=uniform, score=0.745, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=6, knn__p=2, knn__weights=uniform, score=0.658, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=6, knn__p=2, knn__weights=distance, score=0.760, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=6, knn__p=2, knn__weights=distance, score=0.777, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=6, knn__p=2, knn__weights=distance, score=0.728, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=6, knn__p=3, knn__weights=uniform, score=0.715, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=6, knn__p=3, knn__weights=uniform, score=0.739, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=6, knn__p=3, knn__weights=uniform, score=0.680, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=6, knn__p=3, knn__weights=distance, score=0.761, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=6, knn__p=3, knn__weights=distance, score=0.772, total=   0.0s
## [CV] knn__n_neighbors=6, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=6, knn__p=3, knn__weights=distance, score=0.729, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=7, knn__p=1, knn__weights=uniform, score=0.685, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=7, knn__p=1, knn__weights=uniform, score=0.752, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=7, knn__p=1, knn__weights=uniform, score=0.632, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=7, knn__p=1, knn__weights=distance, score=0.745, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=7, knn__p=1, knn__weights=distance, score=0.795, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=7, knn__p=1, knn__weights=distance, score=0.719, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=7, knn__p=2, knn__weights=uniform, score=0.715, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=7, knn__p=2, knn__weights=uniform, score=0.752, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=2, knn__weights=uniform ..............
```

```
## [CV]  knn__n_neighbors=7, knn__p=2, knn__weights=uniform, score=0.678, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=7, knn__p=2, knn__weights=distance, score=0.756, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=7, knn__p=2, knn__weights=distance, score=0.782, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=7, knn__p=2, knn__weights=distance, score=0.734, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=7, knn__p=3, knn__weights=uniform, score=0.696, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=7, knn__p=3, knn__weights=uniform, score=0.730, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=7, knn__p=3, knn__weights=uniform, score=0.697, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=7, knn__p=3, knn__weights=distance, score=0.745, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=7, knn__p=3, knn__weights=distance, score=0.765, total=   0.0s
## [CV] knn__n_neighbors=7, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=7, knn__p=3, knn__weights=distance, score=0.734, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=8, knn__p=1, knn__weights=uniform, score=0.673, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=8, knn__p=1, knn__weights=uniform, score=0.748, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=8, knn__p=1, knn__weights=uniform, score=0.617, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=8, knn__p=1, knn__weights=distance, score=0.740, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=8, knn__p=1, knn__weights=distance, score=0.790, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=8, knn__p=1, knn__weights=distance, score=0.706, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=8, knn__p=2, knn__weights=uniform, score=0.722, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=8, knn__p=2, knn__weights=uniform, score=0.749, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=8, knn__p=2, knn__weights=uniform, score=0.694, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=8, knn__p=2, knn__weights=distance, score=0.760, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=8, knn__p=2, knn__weights=distance, score=0.779, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=8, knn__p=2, knn__weights=distance, score=0.741, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=8, knn__p=3, knn__weights=uniform, score=0.678, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=8, knn__p=3, knn__weights=uniform, score=0.740, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=8, knn__p=3, knn__weights=uniform, score=0.681, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=8, knn__p=3, knn__weights=distance, score=0.731, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=8, knn__p=3, knn__weights=distance, score=0.772, total=   0.0s
## [CV] knn__n_neighbors=8, knn__p=3, knn__weights=distance .............
```

```
## [CV]  knn__n_neighbors=8, knn__p=3, knn__weights=distance, score=0.727, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=9, knn__p=1, knn__weights=uniform, score=0.672, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=9, knn__p=1, knn__weights=uniform, score=0.743, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=1, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=9, knn__p=1, knn__weights=uniform, score=0.612, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=9, knn__p=1, knn__weights=distance, score=0.740, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=9, knn__p=1, knn__weights=distance, score=0.785, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=1, knn__weights=distance .............
## [CV]  knn__n_neighbors=9, knn__p=1, knn__weights=distance, score=0.700, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=9, knn__p=2, knn__weights=uniform, score=0.714, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=9, knn__p=2, knn__weights=uniform, score=0.755, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=2, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=9, knn__p=2, knn__weights=uniform, score=0.685, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=9, knn__p=2, knn__weights=distance, score=0.753, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=9, knn__p=2, knn__weights=distance, score=0.782, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=9, knn__p=2, knn__weights=distance, score=0.734, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=9, knn__p=3, knn__weights=uniform, score=0.683, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=9, knn__p=3, knn__weights=uniform, score=0.754, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=3, knn__weights=uniform ..............
## [CV]  knn__n_neighbors=9, knn__p=3, knn__weights=uniform, score=0.683, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=9, knn__p=3, knn__weights=distance, score=0.729, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=9, knn__p=3, knn__weights=distance, score=0.778, total=   0.0s
## [CV] knn__n_neighbors=9, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=9, knn__p=3, knn__weights=distance, score=0.728, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=1, knn__weights=uniform .............
## [CV]  knn__n_neighbors=10, knn__p=1, knn__weights=uniform, score=0.663, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=1, knn__weights=uniform .............
## [CV]  knn__n_neighbors=10, knn__p=1, knn__weights=uniform, score=0.737, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=1, knn__weights=uniform .............
## [CV]  knn__n_neighbors=10, knn__p=1, knn__weights=uniform, score=0.608, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=1, knn__weights=distance ............
## [CV]  knn__n_neighbors=10, knn__p=1, knn__weights=distance, score=0.736, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=1, knn__weights=distance ............
## [CV]  knn__n_neighbors=10, knn__p=1, knn__weights=distance, score=0.780, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=1, knn__weights=distance ............
## [CV]  knn__n_neighbors=10, knn__p=1, knn__weights=distance, score=0.695, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=2, knn__weights=uniform .............
## [CV]  knn__n_neighbors=10, knn__p=2, knn__weights=uniform, score=0.715, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=2, knn__weights=uniform .............
## [CV]  knn__n_neighbors=10, knn__p=2, knn__weights=uniform, score=0.748, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=2, knn__weights=uniform .............
```

```
## [CV]  knn__n_neighbors=10, knn__p=2, knn__weights=uniform, score=0.687, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=10, knn__p=2, knn__weights=distance, score=0.753, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=10, knn__p=2, knn__weights=distance, score=0.778, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=2, knn__weights=distance .............
## [CV]  knn__n_neighbors=10, knn__p=2, knn__weights=distance, score=0.732, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=3, knn__weights=uniform .............
## [CV]  knn__n_neighbors=10, knn__p=3, knn__weights=uniform, score=0.678, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=3, knn__weights=uniform .............
## [CV]  knn__n_neighbors=10, knn__p=3, knn__weights=uniform, score=0.740, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=3, knn__weights=uniform .............
## [CV]  knn__n_neighbors=10, knn__p=3, knn__weights=uniform, score=0.675, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=10, knn__p=3, knn__weights=distance, score=0.724, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=10, knn__p=3, knn__weights=distance, score=0.768, total=   0.0s
## [CV] knn__n_neighbors=10, knn__p=3, knn__weights=distance .............
## [CV]  knn__n_neighbors=10, knn__p=3, knn__weights=distance, score=0.724, total=   0.0s
## GridSearchCV(cv=3,
##              estimator=Pipeline(steps=[('standardize', StandardScaler()),
##                                        ('knn',
##                                         KNeighborsRegressor(n_jobs=-1))]),
##              param_grid={'knn__n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
##                          'knn__p': [1, 2, 3],
##                          'knn__weights': ['uniform', 'distance']},
##              verbose=3)
##
## [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
## [Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:    0.0s
## [Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.1s remaining:    0.0s
## [Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed:    2.7s finished
```

```
knn_reg_gs.best_estimator_
```

```
## Pipeline(steps=[('standardize', StandardScaler()),
##                 ('knn',
##                  KNeighborsRegressor(n_jobs=-1, n_neighbors=2,
##                                      weights='distance'))])
```

```
test_preds = knn_reg_gs.best_estimator_.predict(X_test)
```

```
from sklearn.metrics import mean_squared_error
results = pd.DataFrame(columns=["Model", "k", "p", "weights","MSE"])
results.loc[len(results)] = ["KNN Regressor Test", knn_reg_gs.best_params_['knn__n_neighbors'],
                             knn_reg_gs.best_params_['knn__p'], knn_reg_gs.best_params_['knn__weights']
                             round(mean_squared_error(y_test, test_preds),3)]
results
```

```
##                 Model  k  p   weights     MSE
## 0  KNN Regressor Test  2  2  distance  15.688
```