hprice.py

```
# importing required packages/libraries/modules/functions

import numpy as np
import pandas as pd
import os
import pylab as pl
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsRegressor as NN
from sklearn.metrics import mean_absolute_error, make_scorer
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.decomposition import PCA

# setting current working directory where input csv is available
# print(os.getcwd())
os.chdir("G:\\Python\\files_cwd")
# print(os.getcwd())

# creating dataframe containing 1405 rows of house price data of London in 1990
hp = pd.read_csv('hpdemo.csv', dtype=float)

# Data Scaling
x_scaler = StandardScaler()
x_scaler.fit(hp[['east','north','fl_area']])
X = x_scaler.transform(hp[['east','north','fl_area']])
# to print first 5 rows and all columns (easting, northing and fl_Area) after standard Z-transformation
# print(X[:5, :])

# KNN Algorithm with k=6, distance metrics as Euclidian and uniform averaging method
reg_object = NN(n_neighbors=6,weights='uniform',p=2)
price = hp['price']/1000.00
reg_object.fit(X,price)

# predicting price (in thousand GBP) of house with easting = 523800, northing = 179750.5 and floor area
of 55 sqm)
predicted_price = reg_object.predict(x_scaler.transform([[523800.0, 179750.5, 55.0]]))
print("Predicted Price without using KNN cross-validation:", predicted_price)
```

```
# Using Mean Absolute Error (MAE) as scoring method for Cross Validated KNN
mae = make_scorer(mean_absolute_error, greater_is_better=False)

opt_nn = GridSearchCV(
            estimator=NN(), scoring=mae,
            param_grid={'n_neighbors': range(1, 35), 'weights': ['uniform', 'distance'], 'p': [1, 2]}
          )

opt_nn.fit(X, price)
predicted_hp = opt_nn.predict(x_scaler.transform([[523800.0, 179750.5, 55.0]]))
print("Predicted Price using KNN cross-validation:", predicted_hp)
print("Parameters for Best Model:", opt_nn.best_estimator_.get_params())
print("Best Score is:",opt_nn.best_score_)

# everything in a neat summary using function
def print_summary(opt_reg_object):
  params = opt_reg_object.best_estimator_.get_params()
  score = - opt_reg_object.best_score_
  print("Nearest Neighbors: %8d" % params['n_neighbors'])
  print("Minkowski P: %8d" % params['p'])
  print("Weighting: %8s" % params['weights'])
  print("MAE Score: %8.2f" % score)
  return

print("Summary wthout pipeline:")
print_summary(opt_nn)

# same thing using pipeline
pipe = Pipeline([('zscores', StandardScaler()),('NNreg', NN(n_neighbors=6, weights='uniform', p=2))])
pipe.fit(hp[['east','north','fl_area']],price)
# without using KNN cross-validation
print(pipe.predict([[523800.0,179750.0,55.0]]))
pipe = Pipeline([('zscores',StandardScaler()),('NNreg',NN())])

opt_nn2 = GridSearchCV(
            estimator = pipe,
            scoring = mae,param_grid = {
                         'NNreg__n_neighbors':range(1,35),
                         'NNreg__weights':['uniform','distance'],
                         'NNreg__p':[1,2]
                         }
```

```
                    )

opt_nn2.fit(hp[['east','north','fl_area']],price)
# Using KNN cross-validation
print(opt_nn2.predict([[523800.0, 179750.0, 55.0]]))

def print_summary2(opt_pipe_object):
    params = opt_pipe_object.best_estimator_.get_params()
    score = - opt_pipe_object.best_score_
    print ("Nearest neighbors: %8d" % params['NNreg__n_neighbors'])
    print ("Minkowski p : %8d" % params['NNreg__p'])
    print ("Weighting : %8s" % params['NNreg__weights'])
    print ("MAE Score : %8.2f" % score)
    return

print("Summary using pipeline:")
print_summary2(opt_nn2)

# Visualisations
east_mesh, north_mesh = np.meshgrid(np.linspace(505000, 555800, 100),np.linspace(158400, 199900,
100))
fl_mesh = np.zeros_like(east_mesh)
fl_mesh[:,:] = np.mean(hp['fl_area'])

print(east_mesh.shape)
print(north_mesh.shape)

grid_predictor_vars = np.array([east_mesh.ravel(),north_mesh.ravel(), fl_mesh.ravel()]).T
hp_pred = opt_nn2.predict(grid_predictor_vars)
hp_mesh = hp_pred.reshape(east_mesh.shape)

fig = pl.figure()
ax = Axes3D(fig)
ax.plot_surface(east_mesh, north_mesh, hp_mesh, rstride=1,cstride=1, cmap='YlOrBr', lw=0.01)
ax.set_xlabel('Easting')
ax.set_ylabel('Northing')
ax.set_zlabel('Price at Mean {} Floor Area'.format(round(np.mean(hp['fl_area']),2)))
pl.show()

# function accepts model prepared using ML pipeline and area to make a 3D-plot
def surf3d(pipe_model,fl_area):
    east_mesh, north_mesh = np.meshgrid(
    np.linspace(505000,555800,100),
    np.linspace(158400,199900,100))
    fl_mesh = np.zeros_like(east_mesh)
    fl_mesh[:,:] = fl_area
    grid_predictor_vars = np.array([east_mesh.ravel(),
    north_mesh.ravel(),fl_mesh.ravel()]).T
    hp_pred = pipe_model.predict(grid_predictor_vars)
    hp_mesh = hp_pred.reshape(east_mesh.shape)
    fig = pl.figure()
    ax = Axes3D(fig)
    ax.plot_surface(east_mesh, north_mesh, hp_mesh, rstride=1, cstride=1, cmap='YlOrBr',lw=0.01)
    ax.set_xlabel('Easting')
    ax.set_ylabel('Northing')
    ax.set_zlabel('Price at {} sqm floor area'.format(fl_area))
    return

# 3D plot for house prices with floor area as 75 sq meter
surf3d(opt_nn2, 75.0)
pl.show()

# 3D plot for house prices with floor area as 125 sq meter
surf3d(opt_nn2, 125.0)
pl.show()

# modifying pipeline to add PCA
pipe = Pipeline([('zscores',StandardScaler()),('prcomp',PCA()),('NNreg',NN())])

opt_nn3 = GridSearchCV(
            estimator = pipe,
            scoring = mae,
            param_grid = {
                    'NNreg__n_neighbors':range(1,35),
                    'NNreg__weights':['uniform','distance'],
                    'NNreg__p':[1,2],
                    'prcomp__n_components':[1,2,3]
                    }
            )

opt_nn3.fit(hp[['east','north','fl_area']],price)
print(opt_nn3.best_estimator_.get_params()['prcomp__n_components'])
print(opt_nn3.best_score_)
```

```
"""
OUTPUT

Predicted Price without using KNN cross-validation: [128.5]
Predicted Price using KNN cross-validation: [121.79031877]
Parameters for Best Model: {'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params':
None, 'n_jobs': 1, 'n_neighbors': 13, 'p': 1, 'weights': 'distance'}
Best Score is: -26.487173782084888
Summary wthout pipeline:
Nearest Neighbours:      13
Minkowski P:       1
Weighting: distance
MAE Score:    26.49
[128.5]
[121.78783506]
Summary using pipeline:
Nearest neighbours:      13
Minkowski p :       1
Weighting : distance
MAE Score :    26.47
(100, 100)
(100, 100)
3
-26.603830189496502
"""
```