In [1]:
```
import numpy as np
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
import pandas as pd
```

In [2]:
```
np.random.seed(598)
x1, x2, x3, x4 = np.random.randn(1000), np.random.randn(1000), np.random.rar
epsilon = np.random.randn(1000)
y = x1 + 2*x2 - x3 + epsilon
```

In [3]:
```
df = pd.DataFrame({"x1": x1, "x2": x2, "x3": x3, "x4": x4, "y": y})
x_set, y_set = df[["x1", "x2", "x3", "x4"]].values, df[["y"]].values
x_train, x_test = x_set[:500], x_set[500:]
y_train, y_test = y_set[:500], y_set[500:]
```

In [4]:
```
def builtin_knn(n):
    regressor = KNeighborsRegressor(n_neighbors=n)
    regressor.fit(x_train, y_train)
    return mean_squared_error(y_test, regressor.predict(x_test))
```

In [5]:
```
print(builtin_knn(4))
print(builtin_knn(5))
```

```
1.5400756698787827
1.3978324747917983
```

In [6]:
```
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1-x2)**2))
```

In [7]:
```
def myKNN(xtrain, ytrain, xtest, k):
    ytest = []

    for x in xtest:
        distances_arr = [euclidean_distance(x, train) for train in xtrain]
        nearest_indices = np.argsort(distances_arr)[:k]
        nearest_neighbors = [ytrain[i] for i in nearest_indices]
        ytest.append(np.mean(nearest_neighbors))

    return ytest
```

In [12]:
```
def mse(test, pred):
    n = len(test)
    error = 0
    for i in range(n):
        error += ((test[i]-pred[i])**2)

    return (float)(error.item() / n)
```

In [13]:
```
ypred4 = myKNN(x_train, y_train, x_test, 4)
print(mse(y_test, ypred4))
ypred5 = myKNN(x_train, y_train, x_test, 5)
print(mse(y_test, ypred5))
```

```
1.540075669878783
1.3978324747917972
```