سوال شماره ۲

در این تمرین قصد داریم با الگوریتم k-Nearest Neighbors از جمله نحوه کارکرد و نحوه پیاده سازی آن از ابتدا در پایتون آشنا شوید. الگوریتم k-Nearest Neighbors یا به اختصار KNN یک تکنیک بسیار ساده است. کل دیتاست به صورت ذخیره شده وجود دارد. هنگامی که یک پیش‌بینی مورد نیاز است، رکوردهای شبیه به k با یک رکورد جدید از دیتاست مکان یابی میشوند. از این همسایگان، یک پیش‌بینی خلاصه انجام می شود.

شباهت بین رکوردها را می توان به روش های مختلف اندازه گیری کرد. به طور کلی، با داده های جدولی، یک نقطه شروع خوب استفاده از فاصله اقلیدسی است. هنگامی که همسایگان کشف شدند، می توان پیش بینی خلاصه را با برگرداندن رایج ترین نتیجه یا گرفتن میانگین انجام داد. به این ترتیب، KNN می‌تواند برای مشکلات طبقه بندی یا رگرسیون استفاده شود.

سه مرحله ی این الگوریتم عبارت است از:

۱- محاسبه ی فاصله اقلیدسی (Euclidean Distance = sqrt(sum i to N (x1_i – x2_i)^2))

۲- یافتن نزدیکترین همسایه ها

۳- پیش بینی کردن و حدس

قطعه کد این برنامه به صورت زیر می باشد.

۱- افزودن کتابخانه های مورد نیاز

```
1  # Make Predictions with k-nearest neighbors on the Amazon servers Dataset
2  from csv import reader
3  from math import sqrt
```

۲- لود کردن فایل csv

```
5  # Load a CSV file
6  def load_csv(filename):
7      dataset = list()
8      with open(filename, 'r') as file:
9          csv_reader = reader(file)
10         for row in csv_reader:
11             if not row:
12                 continue
13             dataset.append(row)
14     return dataset
```

۳- تبدیل کردن رشته ها به عدد ممیزی

```
# Convert string column to float
def str_column_to_float(dataset, column):
    for row in dataset:
        row[column] = float(row[column].strip())
```

۴- تبدیل رشته ها به اینت

```
# Convert string column to integer
def str_column_to_int(dataset, column):
    class_values = [row[column] for row in dataset]
    unique = set(class_values)
    lookup = dict()
    for i, value in enumerate(unique):
        lookup[value] = i
        print('[%s] => %d' % (value, i))
    for row in dataset:
        row[column] = lookup[row[column]]
    return lookup
```

۵- یافتن بیشترین و کمترین مقدار در ستون ها

```
# Find the min and max values for each column
def dataset_minmax(dataset):
    minmax = list()
    for i in range(len(dataset[0])):
        col_values = [row[i] for row in dataset]
        value_min = min(col_values)
        value_max = max(col_values)
        minmax.append([value_min, value_max])
    return minmax
```

۶- سورت کردن داده هایی که مقدار بین صفر و یک دارند

```
# Rescale dataset columns to the range 0-1
def normalize_dataset(dataset, minmax):
    for row in dataset:
        for i in range(len(row)):
            row[i] = (row[i] - minmax[i][0]) / (minmax[i][1] - minmax[i][0])
```

۷- محاسبه ی فاصله اقلیدسی بردار متصل کننده ی دونقطه

```
# Calculate the Euclidean distance between two vectors
def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1)-1):
        distance += (row1[i] - row2[i])**2
    return sqrt(distance)
```

۸- یافتن مشابه ترین همسایه ها

```
# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
    distances = list()
    for train_row in train:
        dist = euclidean_distance(test_row, train_row)
        distances.append((train_row, dist))
    distances.sort(key=lambda tup: tup[1])
    neighbors = list()
    for i in range(num_neighbors):
        neighbors.append(distances[i][0])
    return neighbors
```

۹- حدس زدن همسایه ها

```
# Make a prediction with neighbors
def predict_classification(basket, test_row, num_neighbors):
    neighbors = get_neighbors(basket, test_row, num_neighbors)
    output_values = [row[-1] for row in neighbors]
    prediction = max(set(output_values), key=output_values.count)
    return prediction
```

۱۰- حدس زدن همسایه ها با k- نزدیکترین همسایه روی دیتا

```
# Make a prediction with KNN on Amazon Dataset
filename = 'example.csv'
dataset = load_csv(filename)
for i in range(len(dataset[0])-1):
    str_column_to_float(dataset, i)
```

❇ست کردن تعداد همسایگی ها(سرورها) و ...

```
# convert class column to integers
str_column_to_int(dataset, len(dataset[0])-1)
# define model parameter
num_neighbors = 5
# define a new record
row = [5.7,2.9,4.2,1.3]
# predict the label
label = predict_classification(dataset, row, num_neighbors)
print('Data=%s, Predicted: %s' % (row, label))
```