K-Nearest Neighbours

Geschreven door: Matthijs Koelewijn

Uitleg code

Call tree

main

…….Data__init__

…….Data.assign_labels

…….data_min_max

…….Data.normalise

…….analyse__init__

…….analyse.run

…….Analyse.print_progress

…….Analyse.get_result

…….k_nearest_neighbours

…….sort_dict

…….Sorted__init__

…….List__init__
…….Counter__init__
…….Counter_most_common

…….k_nearest_neighbours

…….Analyse.calculate_winrate

…….Analyse.winner

…….Analyse.print_result

…….k_nearest_neighbours

Totaal-uitleg

Het K-Nearest-Neighbors-programma begint in de **main** en roept vervolgens verschillende andere functies aan om de data te initialiseren, labels toe te wijzen, te normaliseren en te analyseren.

In eerste instantie worden drie **Data** objecten geïnitialiseerd. Deze objecten bevatten elk één dataset: "dataset1.csv", "validation1.csv" en "days.csv". Het initialisatieproces resulteert in drie objecten genaamd training_data, validation_data en unclassified_data.

Vervolgens wordt de functie **assign_labels** toegepast op de **Data** objecten training_data en validation_data. Deze functie wijst labels toe die we zelf hebben gedefinieerd, namelijk de seizoenen winter, lente, zomer en herfst. De functie **assign_labels** wordt niet toegepast op het **Data** object unclassified_data, omdat deze dataset "days.csv" bevat als data source, waarop we deze labels nog niet kunnen toepassen.

Om normalisatie op deze **Data** objecten toe te kunnen passen, willen we alle minimale en maximale datapunten van alle datasets bij elkaar hebben. Daarom wordt de functie **data_min_max** uitgevoerd. Binnen deze functie worden alle datasets aan elkaar geplakt in een array via de **numpy.concatenate** functie.

Nu we alle minimale en maximale datapunten hebben, kunnen we normalisatie toepassen op alle drie de **Data** objecten. We roepen de functie **Data.normalise** aan en geven deze minimale en maximale datapunten mee.

Het volgende object dat wordt aangemaakt is analyse. Dit object wordt geïnitialiseerd met de twee genormaliseerde Data-objecten training_data en validation_data, en de range_k. De range_k bevat de lengte van de training dataset.

Het **analyse** object is het belangrijkste object van dit programma omdat het de voorspellingen genereert en de beste k analyseert. Als eerste wordt de functie **analyse.run** aangeroepen. Binnen deze functie wordt herhaaldelijk **analyse.progress**, **analyse.get_result** en **analyse.calculate_winrate** uitgevoerd. De functie **analyse.progress** wordt slechts gebruikt om de voortgang van de analyse voor de gebruiker aan te geven.

Binnen de functie **analyse.get_result** wordt herhaaldelijk de functie **k_nearest_neighbours** aangeroepen voor een gegeven k, het aantal dichtstbijzijnde datapunten dat wordt gebruikt bij het bepalen van de klasse van het nieuwe datapunt. De functie **k_nearest_neighbours** neemt vier argumenten: de dataset (data), de labels van de datapunten in de dataset (labels), een nieuw datapunt om te classificeren (point) en het aantal dichtstbijzijnde datapunten (k). De functie geeft als output de label van het nieuwe datapunt.

De functie berekent eerst de Euclidische afstand tussen het nieuwe datapunt (point) en elk datapunt in de dataset (data). Dit wordt gedaan met behulp van de Euclidische afstandsformule, die de afstand tussen twee punten in een n-dimensionale ruimte berekent. Hierbij is n het aantal kenmerken van de datapunten in de dataset. Deze afstand wordt toegevoegd aan de dictionary distances.

Vervolgens wordt deze dictionary distances gesorteerd van lage naar hoge afstanden door middel van **sorted_dict,** binnen deze functie wordt de ingebouwde functie **sorted** gebruikt, en wordt een lijst gemaakt van de labels van de k dichtstbijzijnde datapunten.

De **Counter** object wordt in dit geval gebruikt om het aantal keren dat een afstand voorkomt in de lisjt te tellen. Het geeft als output een dictionary waarin elk element van de lijst de sleutel is en de waarde het aantal keer dat het element voorkomt. De output van **Counter** gebruiken we om de meest voorkomende elementen in de lijst te vinden met de functie **Counter.most_common**. Deze

functie geeft een lijst van tuples met de elementen in de **Counter** object en het aantal keren dat ze voorkomen, in aflopende volgorde.

Deze lijst van tuples gebruiken we om te checken of er een gelijkspel is tussen de labels van de k dichtstbijzijnde datapunten. Bij een gelijkspel wordt de **k_nearest_neighbours** functie opnieuw aangeroepen met k-1 dichtstbijzijnde datapunten, totdat er geen gelijkspel meer is.

Na **analyse.get_result** wordt **analyse.calculate_winrate** aangeroepen om de winrate te bepalen. Na alle iteraties van binnen run wordt de winner bepaald aan de hand van de functie **analyse.winner.** Deze functie checkt welke k het beste was aan de hand van de hoogste winrate.

Na het analyseren van de data, wordt de **Analyse.print_result** functie aangeroepen om de voorspelde winnaar te printen naar de terminal. Ten slotte wordt de **k_nearest_neighbours** functie nog een keer aangeroepen, om de beste k die als resultaat kwam uit de analyse te gebruiken voor het genereren van labels voor de unclassified_data **Data** object.

Resultaten

```
Progress: [=========>]  94.25%
Progress: [=========>]  94.52%
Progress: [=========>]  94.79%
Progress: [=========>]  95.07%
Progress: [=========>]  95.34%
Progress: [=========>]  95.62%
Progress: [=========>]  95.89%
Progress: [=========>]  96.16%
Progress: [=========>]  96.44%
Progress: [=========>]  96.71%
Progress: [=========>]  96.99%
Progress: [=========>]  97.26%
Progress: [=========>]  97.53%
Progress: [=========>]  97.81%
Progress: [=========>]  98.08%
Progress: [=========>]  98.36%
Progress: [=========>]  98.63%
Progress: [=========>]  98.9%
Progress: [=========>]  99.18%
Progress: [=========>]  99.45%
Progress: [=========>]  99.73%
Progress: [==========]  100.0%
The best winrate of 64.0% is gained with a k of 75.
The KNN algorithm guesses these labels for file days.csv :
['lente', 'winter', 'lente', 'zomer', 'lente', 'zomer', 'winter', 'winter', 'zomer']

[Done] exited with code=0 in 55.581 seconds
```

In bovenstaande screenshot zien we de output. Hieruit kan opgemaakt worden welke labels zijn geraden voor de ongeclassificeerde data in “days.csv”. En welke winrate de beste is en welke k daarbij hoort. De conclusie kan getrokken worden dat een redelijk accurate voorspelling is gemaakt op basis van de beschikbare gegevens.