K-Nearest Neighbors (KNN) Classifier Example, Study Case: Road Pattern Recognition

Henri Setyo Pambudi

Overview

This script implements a road pattern recognition system using the k-Nearest Neighbors (KNN) algorithm. It takes a dataset of GPS coordinates representing road paths, scales the data, trains a KNN model, and visualizes the results with decision boundaries. Additionally, the model's accuracy is calculated and displayed.

Code

The code is written in Python. Below is the complete implementation:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler

# Input road path as an array of GPS coordinates
road_path = np.array([
    [37.7749, -122.4194],
    [37.5240, -121.5180],
    [37.9880, -121.3207],
    [37.3000, -122.0087],
    ...
])

# Prepare feature matrix
X = np.vstack([road_path])

# Assign labels (dummy labels for demonstration purposes)
y = np.array([1] * len(road_path))

# Scale the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

```

```
# Train KNN classifier
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_scaled, y)

# Calculate model accuracy
accuracy = knn.score(X_scaled, y)
print(f"Accuracy of KNN model: {accuracy * 100:.2f}%")

# Create grid for visualization
x_min, x_max = X_scaled[:, 0].min() - 0.1, X_scaled[:, 0].
    max() + 0.1
y_min, y_max = X_scaled[:, 1].min() - 0.1, X_scaled[:, 1].
    max() + 0.1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.001),
                     np.arange(y_min, y_max, 0.001))

# Predict decision boundary
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plotting
plt.figure(figsize=(10, 8))
plt.contourf(xx, yy, Z, alpha=0.3, cmap='summer')
plt.scatter(road_path[:, 0], road_path[:, 1], c='green',
    label='On the road', edgecolors='black')

plt.title("Road Pattern Recognition with KNN")
plt.xlabel("Latitude")
plt.ylabel("Longitude")
plt.legend(loc="best")
plt.show()
```

Detail Information:

Input Data

The variable `road_path` contains a list of GPS coordinates representing road points. These are stored in a NumPy array.

Feature Scaling

The `StandardScaler` from `scikit-learn` is used to standardize the features (latitude and longitude) to have zero mean and unit variance.

KNN Classifier

The k-Nearest Neighbors algorithm is implemented using `KNeighborsClassifier`. In this example, the number of neighbors (`n_neighbors`) is set to 10.

Accuracy Calculation

The model's accuracy is calculated using the `score` method of the classifier, which compares predictions with the true labels.

Visualization

A decision boundary is created by predicting values for a mesh grid of latitude and longitude. The `contourf` method is used to plot the decision boundary, while the road points are visualized with green markers.

Output

The script outputs:

- The accuracy of the KNN model as a percentage.
- A plot visualizing the road path and the decision boundary.

Dependencies

The script requires the following Python libraries:

- `numpy`
- `matplotlib`
- `scikit-learn`

Usage

This code is demonstration of my thesis as Bachelor Student Final Task. K-Nearest Neighbors algorithm for road pattern recognition, working with geographical data (latitude and longitude as **dummy data**). I want to use it to recognize road so I can adjust another code to give suggestion on driver of vehicle. The road is recognized with *neighbors*: values that still classified as green track to display opmized speed for suggestion using machine learning.