Classification with K-Nearest Neighbors (KNN) using Scikit-Learn

---

Project Overview

This project demonstrates how to implement a **K-Nearest Neighbors (KNN) Classifier** using Python's Scikit-Learn library. KNN is a simple, instance-based learning algorithm that classifies data points based on the majority class among their 'k' nearest neighbors in the feature space.

Why Use K-Nearest Neighbors?

- **Simplicity**: KNN is easy to understand and implement.
- **Versatility**: It can be used for both classification and regression tasks.
- **Non-Parametric**: KNN makes no assumptions about the underlying data distribution.
- **Adaptability**: The model can be easily updated with new data without retraining.

Prerequisites

Required Libraries

- `pandas`: For data manipulation and analysis.
- `numpy`: For numerical computations.
- `scikit-learn`: For machine learning algorithms and evaluation metrics.
- `matplotlib` & `seaborn`: For data visualization.

Installation

Install the necessary libraries using pip:

```
pip install pandas numpy scikit-learn matplotlib seaborn
```

Files Included

- `your_dataset.csv`: The dataset file containing the features and target variable.
- `knn_classification.py`: The Python script implementing the KNN Classifier.

Code Description

The implementation is divided into several key steps:

1. Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

2. Loading and Exploring the Dataset

```
 # Load the dataset
data = pd.read_csv('your_dataset.csv')

# Display the first few rows
print(data.head())
```

3. Preprocessing the Data

```
 # Assuming the last column is the target variable
X = data.iloc[:, :-1]  # Features
y = data.iloc[:, -1]   # Target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
```

4. Training the KNN Classifier

```
 # Initialize and train the k-NN classifier
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
```

5. Making Predictions

```
 # Make predictions on the test set
y_pred = knn_model.predict(X_test)
```

6. Evaluating the Model

```
 # Confusion matrix, classification report, and accuracy score
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))
```

7. Visualizing the Confusion Matrix

```
 # Plot confusion matrix
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

Expected Outputs

- **Confusion Matrix**: A table showing the performance of the classification model.
- **Classification Report**: Includes precision, recall, f1-score, and support for each class.
- **Accuracy Score**: The overall accuracy of the model.
- **Confusion Matrix Heatmap**: A visual representation of the confusion matrix.

Use Cases

- **Recommendation Systems**: Suggesting products or content based on user similarity.
- **Image Recognition**: Classifying images based on pixel intensity comparisons.
- **Anomaly Detection**: Identifying unusual patterns that do not conform to expected behavior.
- **Medical Diagnosis**: Predicting diseases based on patient attributes.

Future Enhancements

- **Choosing Optimal 'k'**: Use cross-validation to determine the best number of neighbors.
- **Feature Scaling**: Apply normalization techniques to improve distance calculations.
- **Handling Large Datasets**: Implement efficient algorithms or approximate methods for scalability.
- **Distance Metrics**: Experiment with different distance measures like Manhattan or Minkowski distances.

References

- KNeighborsClassifier — scikit-learn 1.5.2 documentation
- K-Nearest Neighbors (KNN) Classification with scikit-learn | DataCamp
- KNN Classification using Scikit Learn | by Vishakha Ratnakar