K-Nearest Neighbors (K-NN) Classification

Aniakwa Nathan

July 31, 2025

1 Introduction

This document outlines the implementation of a K-Nearest Neighbors (K-NN) classifier to predict purchases based on age and estimated salary from the dataset `Social_Network_Ads.csv`. The workflow includes data preprocessing, model training, evaluation, and analysis.

2 Importing Libraries

The following Python libraries are used:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

3 Importing the Dataset

The dataset is loaded and the first five rows are shown below:

```
df = pd.read_csv("Social_Network_Ads.csv")
df.head()
```

Output (first 5 rows):

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

Features (`Age`, `EstimatedSalary`) and the target (`Purchased`) are extracted:

```
X = df.iloc[:, :-1].values
Y = df.iloc[:, -1].values
```

4 Splitting the Dataset

The dataset is split into 80% training and 20% test sets with a random seed for reproducibility:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
    test_size=0.2, random_state=45)
```

5 Feature Scaling

Features are standardized to ensure K-NN's distance metric is not biased by scale differences:

```
Sc = StandardScaler()
X_train = Sc.fit_transform(X_train)
X_test = Sc.transform(X_test)
```

6 Training the K-NN Model

A K-NN classifier is trained with 5 neighbors using the Euclidean distance metric:

```
classifier = KNeighborsClassifier(n_neighbors=5,
    metric="minkowski", p=2)
classifier.fit(X_train, Y_train)
```

7 Predicting a New Result

The model predicts the outcome for a new input (`Age=30`, `EstimatedSalary=87000`):

```
print(classifier.predict(Sc.transform([[30, 87000]])))
```

Output: `[0]`

8 Predicting the Test Set Results

Predictions are made for the test set and compared with actual values:

```
y_pred = classifier.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred), 1),
    Y_test.reshape(len(Y_test), 1)), 1))
```

Output (partial, first 5 rows):

```
[[1 1]
 [1 1]
 [0 0]
 [1 1]
 [0 0]]
```

9 Making the Confusion Matrix

The model's performance is evaluated using a confusion matrix and accuracy score:

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(Y_test, y_pred)
print(cm)
accuracy_score(Y_test, y_pred)
```

Output:

```
[[44  4]
 [ 4 28]]
```

Accuracy: 0.9 (90%)

10 Analysis

The K-NN classifier achieves a 90% accuracy on the test set, with 44 true negatives, 28 true positives, 4 false positives, and 4 false negatives. The high accuracy suggests effective separation of classes based on age and salary. However, the 8 misclassifications indicate potential for improvement, such as tuning the number of neighbors or exploring feature interactions. The dataset's numerical features and clear decision boundaries make it well-suited for K-NN.