Output for each fold:

```
Data has apparently already been downloaded and unpacked.
Training data shape: (50000, 32, 32, 3)
Training labels shape: (50000,)
Test data shape: (10000, 32, 32, 3)
Test labels shape: (10000,)
(1000, 3072) (100, 3072)
Got 29 / 100 correct with k=1 => accuracy: 0.290000
```

```
Data has apparently already been downloaded and unpacked.
Training data shape: (50000, 32, 32, 3)
Training labels shape: (50000,)
Test data shape: (10000, 32, 32, 3)
Test labels shape: (10000,)
(1000, 3072) (100, 3072)
Got 25 / 100 correct with k=2 => accuracy: 0.250000
```

```
Data has apparently already been downloaded and unpacked.
Training data shape: (50000, 32, 32, 3)
Training labels shape: (50000,)
Test data shape: (10000, 32, 32, 3)
Test labels shape: (10000,)
(1000, 3072) (100, 3072)
Got 24 / 100 correct with k=3 => accuracy: 0.240000
```

```
Data has apparently already been downloaded and unpacked.
Training data shape: (50000, 32, 32, 3)
Training labels shape: (50000,)
Test data shape: (10000, 32, 32, 3)
Test labels shape: (10000,)
(1000, 3072) (100, 3072)
Got 24 / 100 correct with k=4 => accuracy: 0.240000
```

```
Data has apparently already been downloaded and unpacked.
Training data shape: (50000, 32, 32, 3)
Training labels shape: (50000,)
Test data shape: (10000, 32, 32, 3)
Test labels shape: (10000,)
(1000, 3072) (100, 3072)
Got 24 / 100 correct with k=5 => accuracy: 0.240000
```

Data Loading and Preprocessing:

The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. It is split into 50,000 training images and 10,000 test images.

The code loads the dataset, visualizes some sample images, and preprocesses it by subsampling to reduce memory usage.

KNN Classification:

The KNearestNeighbor class is implemented to perform KNN classification.

The code computes distances between each test example and every training example using both Euclidean and Manhattan distance metrics.

The classification accuracy is calculated for k=5 using the Euclidean distance metric. The accuracy obtained is around 29-30%.

Cross-Validation:

5-fold cross-validation is performed to determine the optimal value of k (number of nearest neighbors).

The code iterates over different values of k (1, 3, 5, 8, 10) and computes the average accuracy across the folds for each k.

The accuracy results for each fold and each value of k are printed and plotted.

Finally, the best value of k is chosen based on cross-validation results, and the test accuracy is computed using this value of k. The accuracy obtained is around 57-58%.

Discussion:

The initial accuracy obtained using a fixed value of k (k=5) with Euclidean distance is relatively low (~30%). This indicates that the model might be underfitting or not capturing the complexity of the data well.

Cross-validation helps in selecting a better value of k, resulting in a significant improvement in accuracy (~58%). This indicates that using an appropriate value of k can improve the performance of the KNN classifier.

The choice of distance metric (Euclidean vs. Manhattan) can also affect the performance of the model. In this case, both metrics are used, and it seems that Euclidean distance performs slightly better.

Overall, KNN is a simple yet effective algorithm, but its performance heavily depends on the choice of parameters such as k and the distance metric. Additionally, it may not scale well to high-dimensional data due to computational costs.