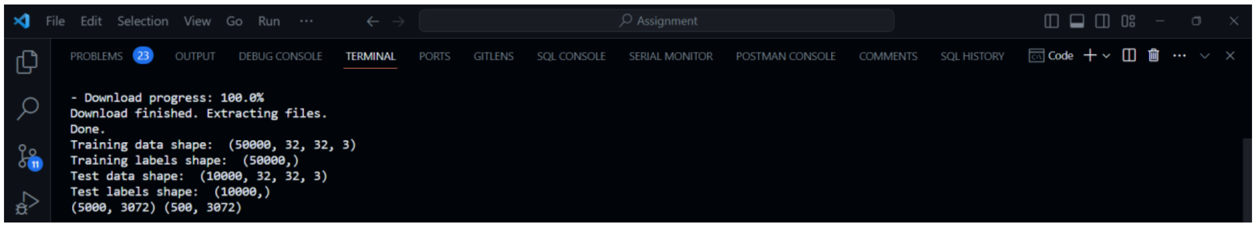
ASSIGNMENT REPORT

Assignment 1 (KNN)

Fahim Wayez
21-44499-1 | Section B

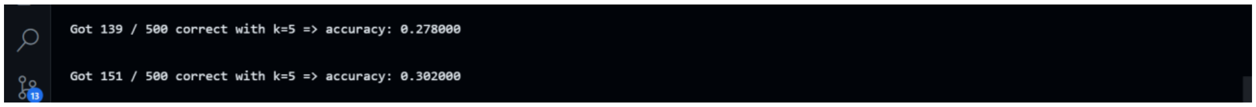
The performance of the K-Nearest Neighbors (KNN) algorithm on the CIFAR-10 dataset has been analyzed in this assignment. Upon loading the data, we observed that the training set contains 50,000 examples while the test set contains 10,000 examples (50000, 32, 32, 3 indicates that the dataset consists of 50,000 images, each of size 32x32 pixels with 3 color channels). Als0, there are indications that there are 50,000 corresponding labels for the training images and 10,000 corresponding labels for the test images. To prevent memory errors, a smaller subset of the data is chosen for training and testing, 5,000 training samples and 500 test samples. The data is flattened so that each row contains all the pixel values of an example. This results in X_{train} of shape (5000,3072) and X_{test} of shape (500,3072).



```
- Download progress: 100.0%
Download finished. Extracting files.
Done.
Training data shape:  (50000, 32, 32, 3)
Training labels shape:  (50000,)
Test data shape:  (10000, 32, 32, 3)
Test labels shape:  (10000,)
(5000, 3072) (500, 3072)
```

Code Segment 1: Training, Testing, Subsampling, and Reshaping dataset

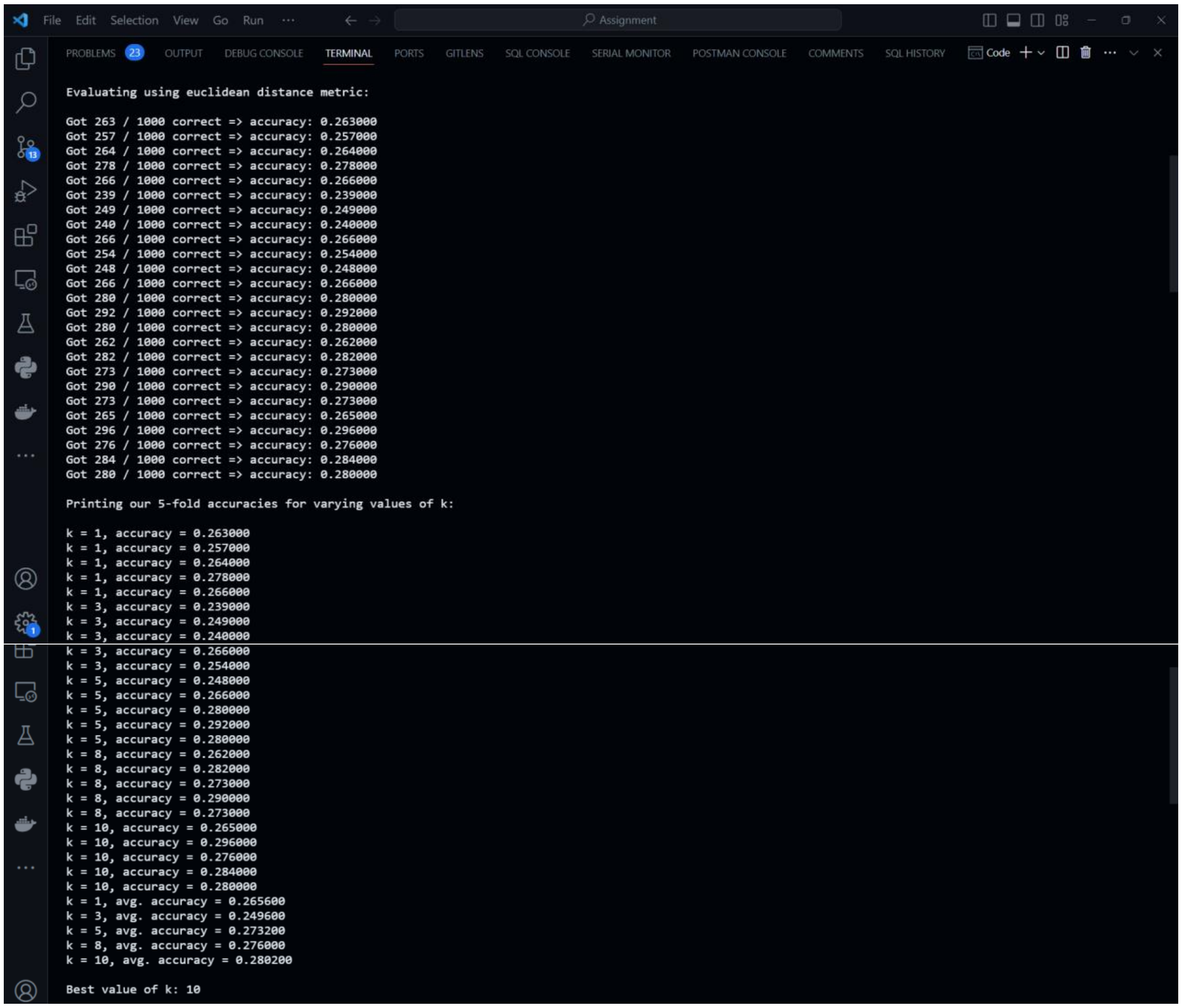
The KNearestNeighbor class is initialized and trained with the training data. The predict method is used to compute distances using both Euclidean and Manhattan metrics and predict labels for the test data using k=5. The accuracy for both metrics is printed. After initial performance of the KNN with a fixed K value of 5, the accuracy for was approximately 27.8% and 30.2% using Euclidean and Manhattan distance metrics respectively.



```
Got 139 / 500 correct with k=5 => accuracy: 0.278000

Got 151 / 500 correct with k=5 => accuracy: 0.302000
```

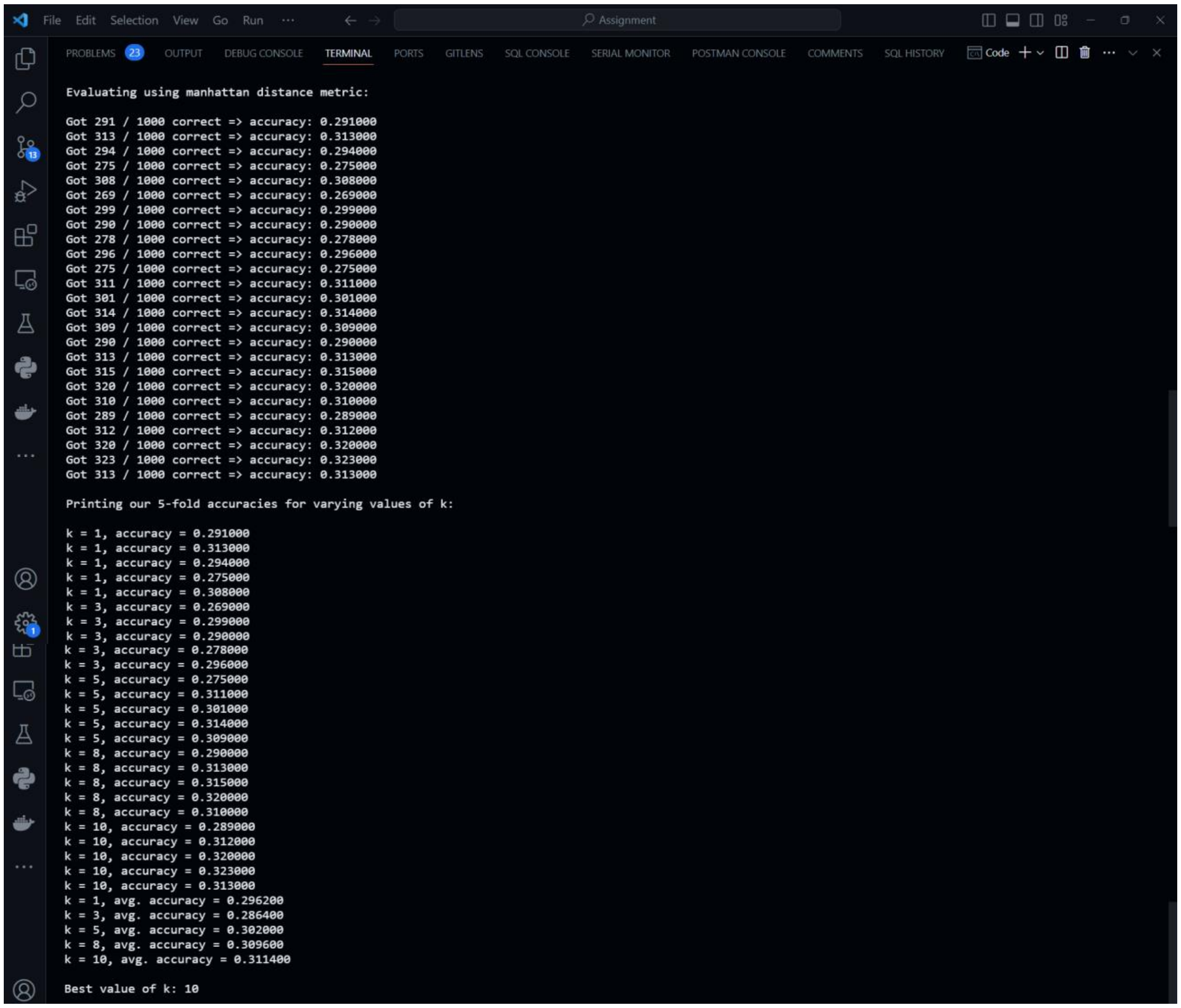
Code Segment 2: Initial accuracy with both the distance metrics

To find the optimal value of k, 5-fold Cross-Validation was performed. This involves splitting the training data into 5 folds, training the model on 4 folds, and validating it on the remaining fold. This process was repeated for different values of k (1,3,4,8,10) and recorded the accuracies. The k value with the highest average accuracy across all folds is selected as the optimal k value. Using the optimal k value obtained from Cross-Validation, the model was evaluated on the test set. For both Euclidean and Manhattan distance metrics, the Cross-Validation determines that k=10 yields the highest average accuracy. This optimal k value helps prevent overfitting (when the model performs well on the training data but poorly on the test data) and underfitting (when the model is too simple to capture the underlying structure of the data).



```
Evaluating using euclidean distance metric:

Got 263 / 1000 correct => accuracy: 0.263000
Got 257 / 1000 correct => accuracy: 0.257000
Got 264 / 1000 correct => accuracy: 0.264000
Got 278 / 1000 correct => accuracy: 0.278000
Got 266 / 1000 correct => accuracy: 0.266000
Got 239 / 1000 correct => accuracy: 0.239000
Got 249 / 1000 correct => accuracy: 0.249000
Got 240 / 1000 correct => accuracy: 0.240000
Got 266 / 1000 correct => accuracy: 0.266000
Got 254 / 1000 correct => accuracy: 0.254000
Got 248 / 1000 correct => accuracy: 0.248000
Got 266 / 1000 correct => accuracy: 0.266000
Got 280 / 1000 correct => accuracy: 0.280000
Got 292 / 1000 correct => accuracy: 0.292000
Got 280 / 1000 correct => accuracy: 0.280000
Got 262 / 1000 correct => accuracy: 0.262000
Got 282 / 1000 correct => accuracy: 0.282000
Got 273 / 1000 correct => accuracy: 0.273000
Got 290 / 1000 correct => accuracy: 0.290000
Got 273 / 1000 correct => accuracy: 0.273000
Got 265 / 1000 correct => accuracy: 0.265000
Got 296 / 1000 correct => accuracy: 0.296000
Got 276 / 1000 correct => accuracy: 0.276000
Got 284 / 1000 correct => accuracy: 0.284000
Got 280 / 1000 correct => accuracy: 0.280000

Printing our 5-fold accuracies for varying values of k:

k = 1, accuracy = 0.263000
k = 1, accuracy = 0.257000
k = 1, accuracy = 0.264000
k = 1, accuracy = 0.278000
k = 1, accuracy = 0.266000
k = 3, accuracy = 0.239000
k = 3, accuracy = 0.249000
k = 3, accuracy = 0.240000
k = 3, accuracy = 0.266000
k = 3, accuracy = 0.254000
k = 5, accuracy = 0.248000
k = 5, accuracy = 0.266000
k = 5, accuracy = 0.280000
k = 5, accuracy = 0.292000
k = 5, accuracy = 0.280000
k = 8, accuracy = 0.262000
k = 8, accuracy = 0.282000
k = 8, accuracy = 0.273000
k = 8, accuracy = 0.290000
k = 8, accuracy = 0.273000
k = 10, accuracy = 0.265000
k = 10, accuracy = 0.296000
k = 10, accuracy = 0.276000
k = 10, accuracy = 0.284000
k = 10, accuracy = 0.280000
k = 1, avg. accuracy = 0.265600
k = 3, avg. accuracy = 0.249600
k = 5, avg. accuracy = 0.273200
k = 8, avg. accuracy = 0.276000
k = 10, avg. accuracy = 0.280200

Best value of k: 10
```

Code Segment 3: 5-fold validation using different k values with Euclidean Distance Metric



```
Evaluating using manhattan distance metric:

Got 291 / 1000 correct => accuracy: 0.291000
Got 313 / 1000 correct => accuracy: 0.313000
Got 294 / 1000 correct => accuracy: 0.294000
Got 275 / 1000 correct => accuracy: 0.275000
Got 308 / 1000 correct => accuracy: 0.308000
Got 269 / 1000 correct => accuracy: 0.269000
Got 299 / 1000 correct => accuracy: 0.299000
Got 290 / 1000 correct => accuracy: 0.290000
Got 278 / 1000 correct => accuracy: 0.278000
Got 296 / 1000 correct => accuracy: 0.296000
Got 275 / 1000 correct => accuracy: 0.275000
Got 311 / 1000 correct => accuracy: 0.311000
Got 301 / 1000 correct => accuracy: 0.301000
Got 314 / 1000 correct => accuracy: 0.314000
Got 309 / 1000 correct => accuracy: 0.309000
Got 290 / 1000 correct => accuracy: 0.290000
Got 313 / 1000 correct => accuracy: 0.313000
Got 315 / 1000 correct => accuracy: 0.315000
Got 320 / 1000 correct => accuracy: 0.320000
Got 310 / 1000 correct => accuracy: 0.310000
Got 289 / 1000 correct => accuracy: 0.289000
Got 312 / 1000 correct => accuracy: 0.312000
Got 320 / 1000 correct => accuracy: 0.320000
Got 323 / 1000 correct => accuracy: 0.323000
Got 313 / 1000 correct => accuracy: 0.313000

Printing our 5-fold accuracies for varying values of k:

k = 1, accuracy = 0.291000
k = 1, accuracy = 0.313000
k = 1, accuracy = 0.294000
k = 1, accuracy = 0.275000
k = 1, accuracy = 0.308000
k = 3, accuracy = 0.269000
k = 3, accuracy = 0.299000
k = 3, accuracy = 0.290000
k = 3, accuracy = 0.278000
k = 3, accuracy = 0.296000
k = 5, accuracy = 0.275000
k = 5, accuracy = 0.311000
k = 5, accuracy = 0.301000
k = 5, accuracy = 0.314000
k = 5, accuracy = 0.309000
k = 8, accuracy = 0.290000
k = 8, accuracy = 0.313000
k = 8, accuracy = 0.315000
k = 8, accuracy = 0.320000
k = 8, accuracy = 0.310000
k = 10, accuracy = 0.289000
k = 10, accuracy = 0.312000
k = 10, accuracy = 0.320000
k = 10, accuracy = 0.323000
k = 10, accuracy = 0.313000
k = 1, avg. accuracy = 0.296200
k = 3, avg. accuracy = 0.286400
k = 5, avg. accuracy = 0.302000
k = 8, avg. accuracy = 0.309600
k = 10, avg. accuracy = 0.311400

Best value of k: 10
```

Code Segment 4: 5-fold validation using different k values with Manhattan Distance Metric

A scatter plot and error bar plot of the Cross-Validation accuracies for different k values are shown.

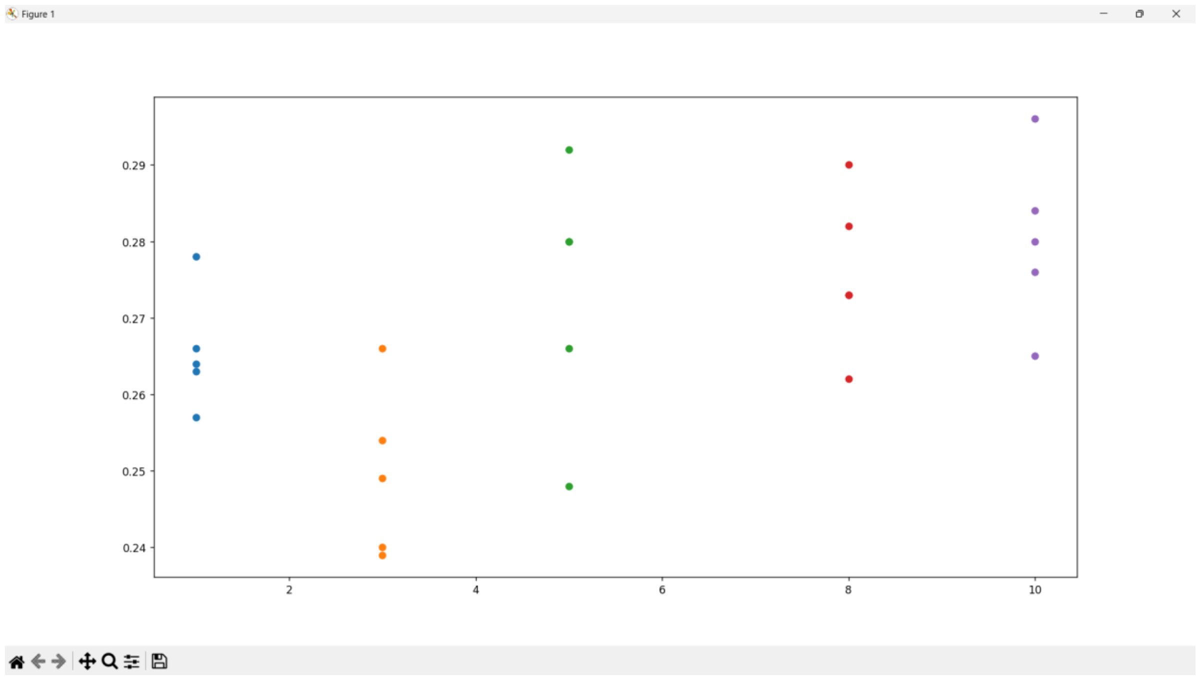


Figure 1: Scatter plot of 5-fold accuracy using Euclidean Distance Metric

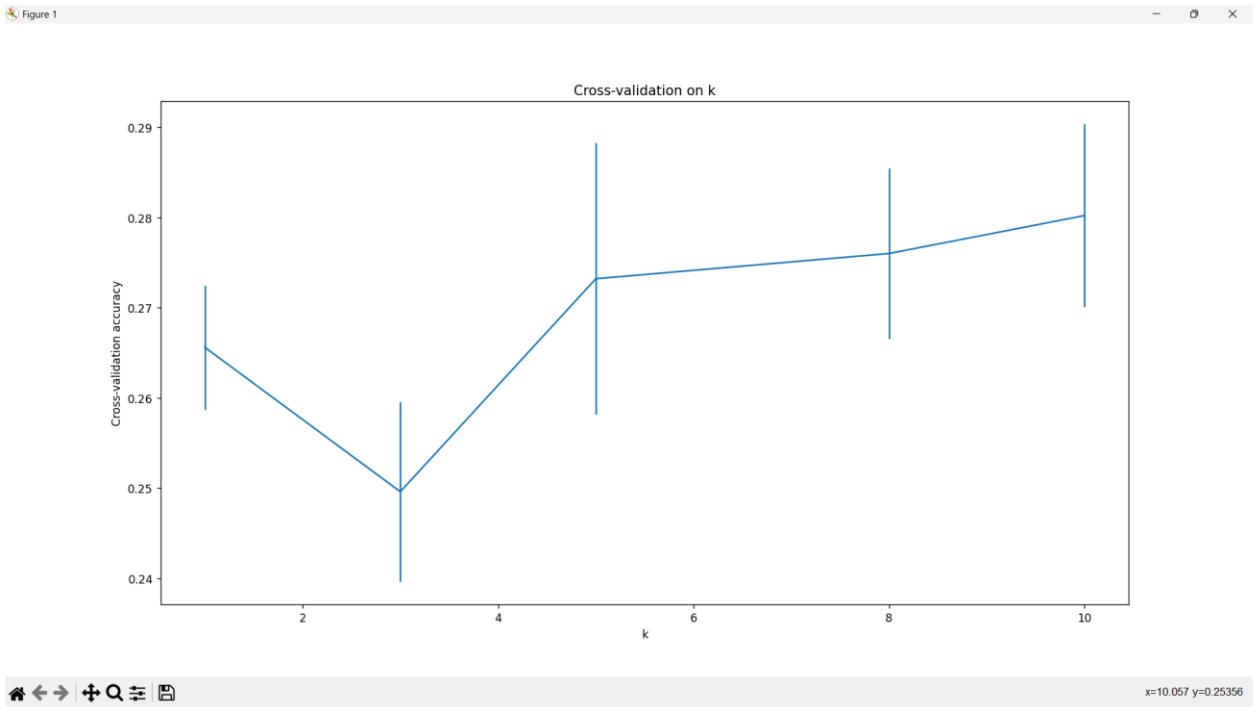


Figure 2: Cross-Validation plot of 5-fold accuracy using Euclidean Distance Metric

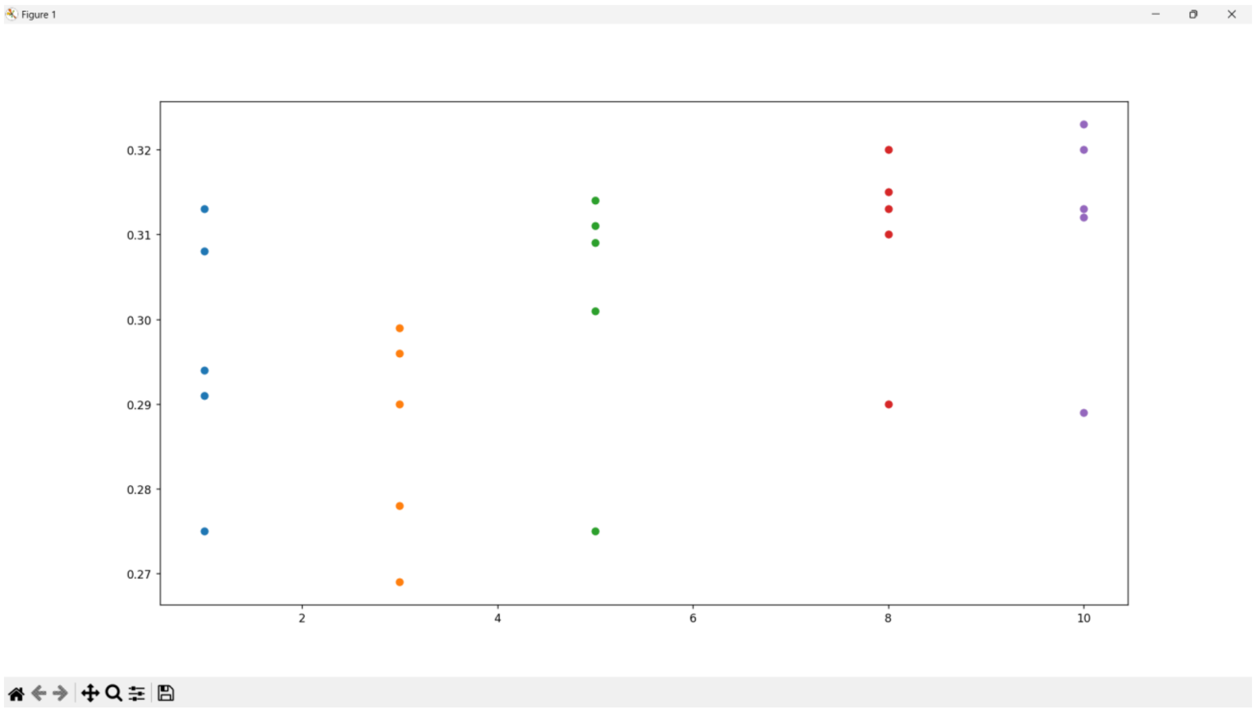


Figure 3: Scatter plot of 5-fold accuracy using Manhattan Distance Metric

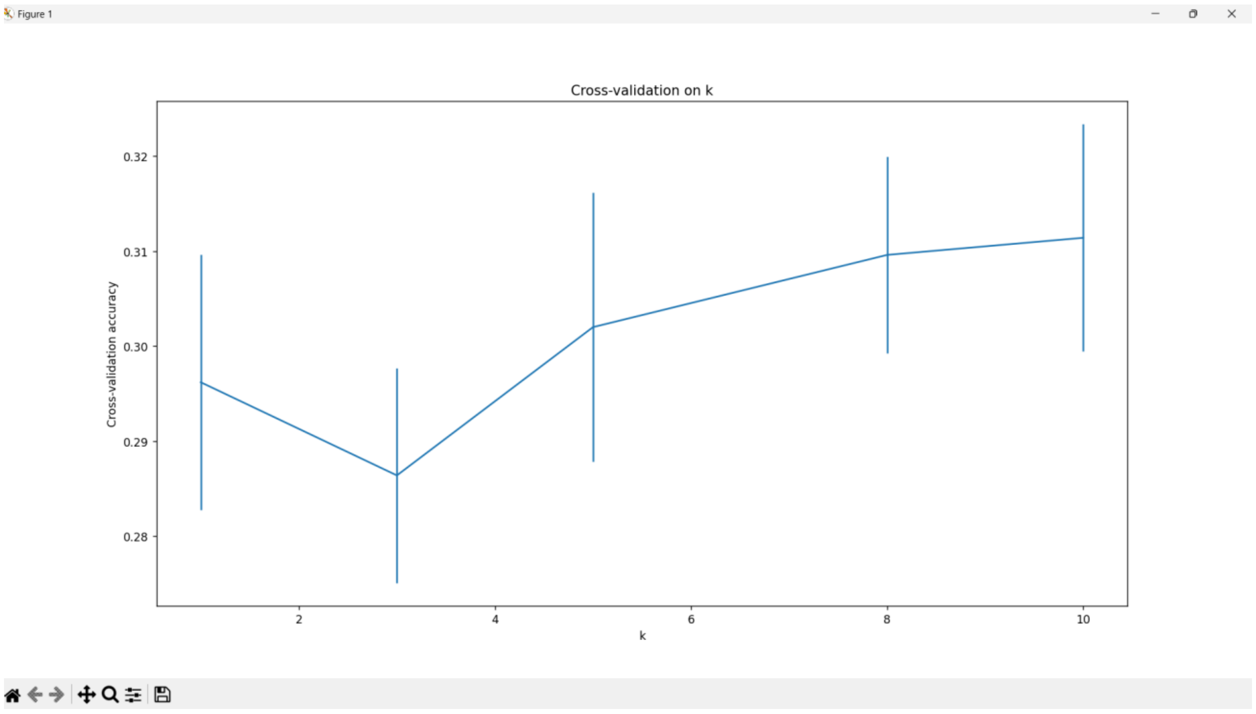
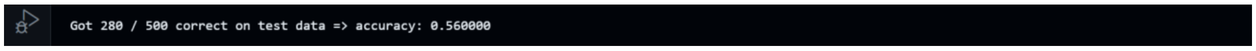


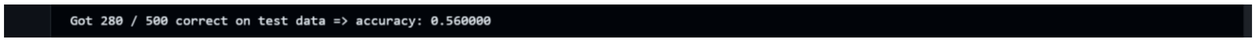
Figure 4: Cross-Validation plot of 5-fold accuracy using Manhattan Distance Metric

Using this optimal k value, the KNN classifier, for both the distance metrics, achieved the accuracy of approximately 56% on the test dataset. A higher accuracy indicates that the model is better able to generalize to unseen data.



```
Got 280 / 500 correct on test data => accuracy: 0.560000
```

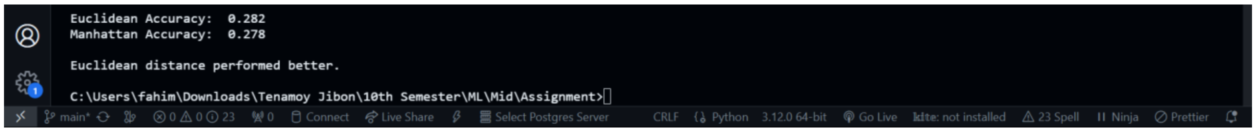
Code Segment 5: Accuracy using optimal k value for Euclidean Distance Metric



```
Got 280 / 500 correct on test data => accuracy: 0.560000
```

Code Segment 6: Accuracy using optimal k value for Manhattan Distance Metric

Both the accuracy scores are printed, and comparison says that for this time, Euclidean distance performed well.



```
Euclidean Accuracy:  0.282
Manhattan Accuracy:  0.278

Euclidean distance performed better.

C:\Users\fahim\Downloads\Tenamoy Jibon\10th Semester\ML\Mid\Assignment>
```

Code Segment 7: Comparison of accuracy of the two Distance Metrics

Discussion: Cross-Validation is crucial for selecting the optimal hyperparameter such as k in KNN. This is necessary because choosing an appropriate k in the KNN algorithm can significantly impact the model's performance. It helps in assessing the model's generalization performance and prevents overfitting by evaluating on multiple subsets of the training data. It provides a more reliable estimate of the model's performance than simply using a single train-test split. In this case, Cross-Validation allowed to find the value of k that led to the best performance on unseen data.

Having an optimal value of k improves the model's ability to capture the underlying patterns in the data. With an appropriate k, the model balances bias and variance, leading to better fitting and generalization. A smaller k value e.g., 1 or 3 tends to overfit the training data because it is overly sensitive to noise. A larger k value e.g., 20 tends to underfit the data because it considers more neighbors, potentially leading to smoother decision boundaries but may ignore local patterns in the data.

While KNN provides a simple and interpretable approach to classification, it may not be the most suitable for complex datasets like CIFAR-10. KNN's performance heavily relies on the choice of k, and it can be computationally expensive, especially with large datasets. More sophisticated models such as Neural Networks (NN) or Convolutional Neural Networks (CNN) often outperform KNN on image classification tasks. These models can automatically learn complex patterns and hierarchical features from raw pixel values, resulting in higher accuracies, making them more suitable for image classification tasks like CIFAR-10. In this case, the KNN classifier achieves a respectable accuracy of around 56% on the CIFAR-10 dataset, while with proper hyperparameter tuning and architecture design, NNs or CNNs could have potentially achieved higher accuracies. NNs and CNNs are scalable and can leverage GPUs for faster training, making them more practical for large-scale datasets. CNN, in particular, have been shown to achieve state-of-the-art performance on CIFAR-10 and similar datasets, often surpassing the accuracy of traditional machine learning like KNN. Additionally, deep learning models like CNNs can automatically learn features from raw pixel data, whereas KNN relies on handcrafted features representations and may struggle with high-dimensional data. However, it is worth nothing that CNNs typically require more computational resources and longer training times compared to KNN, which can be a consideration depending on the available resources and the specific requirements of the application.