American International University- Bangladesh (AIUB)

Faculty of Science & Technology (FST)

Assignment-1: KNN

Course Name:	Machine Learning	**Course Code:**	CSC4232
Semester:	Spring 2024	**Faculty**	**Prof. Dr. Md. Asraf Ali**
Student Name:	**Isham Newaz**	**Student ID:**	**21-45073-2**
Department:	**CSE**	**Section:**	**A**

1. Loading the File and Extracting.

For calculations of we used google collaborator. The CIFAR-10 dataset was uploaded to google drive and it was extracted.

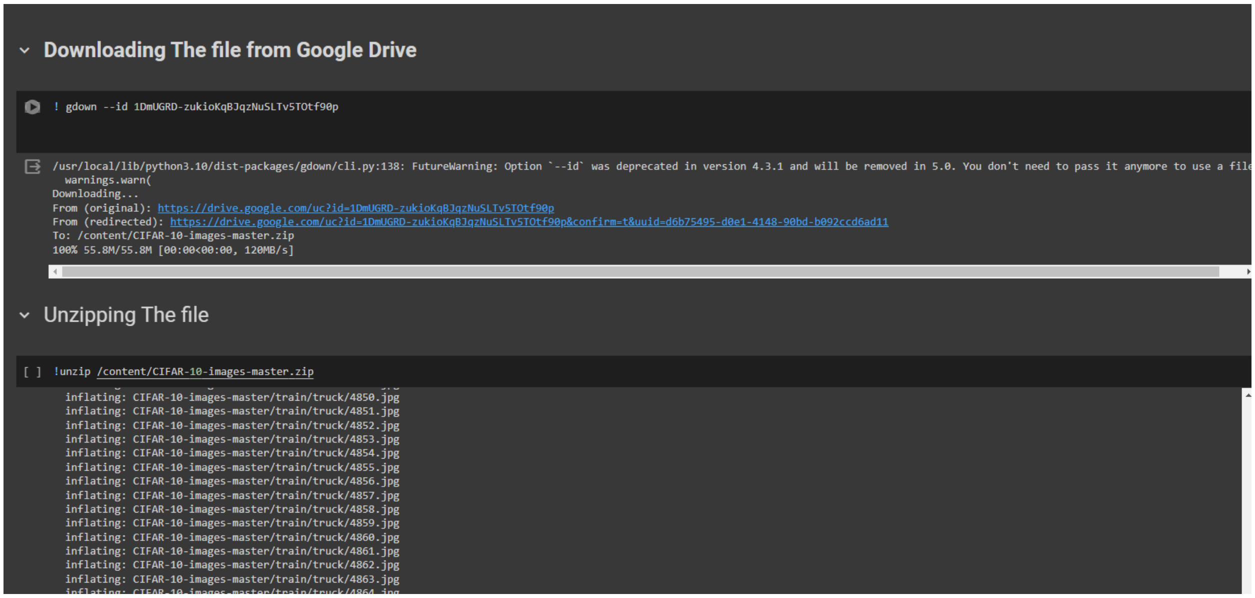

Figure -1: Loading and Extracting Dataset

2. Importing The Library Files

Library Includes:

1. sk-learn
2. cv2
3. matplotlib
4. os
5. glob
6. skimage

Figure -2: Importing Header files

3. Loading/Viewing Image

An Image was loaded to see if the data set file was successfully imported and was extracted

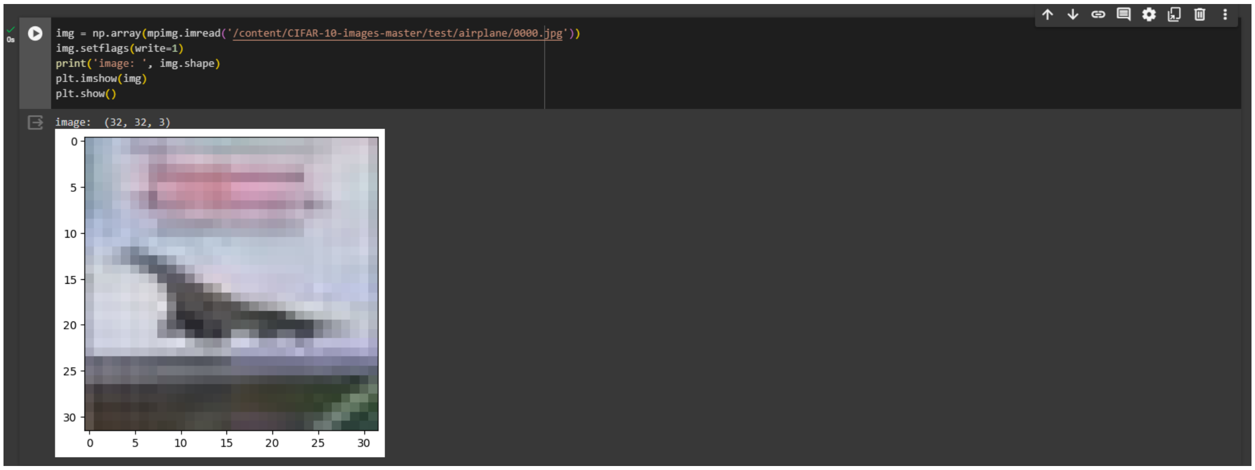

Figure -3: Viewing Image

4. Resizing and Reducing Image quality

The loaded image was resized to 32,32 and the pixel density was reduced to 3 for reducing load form Google Collaboratory server for fast processing.

Figure -4: Viewing Image

5. Converting to Greyscale

The image was converted to Greyscale, and we have extracted to image features. There are 324 features for each image loaded.

```
[4] fd, hog_image=hog(resized_img, visualize=True,multichannel=True)
    print(fd.shape)
    print(fd)
    print(hog_image.shape)
    plt.axis("off")
    plt.imshow(hog_image,cmap="gray")
    plt.show()
```

```
(324,)
[0.12289523 0.07285873 0.02254765 0.00464298 0.02225129 0.04511014
 0.00927719 0.04684367 0.13833722 0.03518563 0.01710244 0.00371876
 0.07335362 0.19788371 0.1041227  0.01864666 0.0399843  0.00838845
 0.02941866 0.01252893 0.01498813 0.05458684 0.25993748 0.10422976
 0.01427526 0.00570432 0.00722109 0.02151719 0.03549209 0.03426088
 0.04550969 0.10863708 0.18902028 0.20535181 0.02664611 0.02690841
 0.         0.         0.01822191 0.05774341 0.22803926 0.20367895
 0.25993748 0.00824058 0.         0.00229948 0.00205672 0.02001865
 0.05833619 0.156271   0.06091585 0.01470475 0.01256734 0.
 0.06059559 0.01149973 0.00443692 0.02048532 0.25544602 0.0448577
 0.00727627 0.03340314 0.03009221 0.07266811 0.09957069 0.08148263
 0.11687239 0.25993748 0.25993748 0.22680922 0.12828171 0.15833352
 0.0068767  0.01645871 0.00245352 0.17907423 0.25993748 0.25993748
 0.1236546  0.00411344 0.         0.03746293 0.01820935 0.00395945
 0.07810124 0.21069123 0.11086177 0.01985351 0.04257218 0.00893137
 0.03132271 0.01333983 0.0159582  0.05811983 0.26886061 0.11097576
 0.01519919 0.00607351 0.00768845 0.07671817 0.04622507 0.01844133
 0.01430681 0.04900294 0.00163534 0.04090826 0.03134913 0.05608995
 0.         0.         0.01940127 0.06148071 0.24279851 0.21686155
 0.26886061 0.00877393 0.         0.00244831 0.00218984 0.0213143
 0.06211184 0.16638524 0.06485847 0.01565648 0.01338073 0.
 0.01461361 0.01325119 0.02701248 0.05232017 0.05081072 0.01794468
 0.00323286 0.02215934 0.02811902 0.07737136 0.10601515 0.08675639
 0.12443665 0.26886061 0.26886061 0.24148886 0.13658441 0.16858125
 0.00732178 0.01752395 0.00261232 0.19066435 0.26886061 0.26886061
 0.13165783 0.00437967 0.         0.         0.01997428 0.1018928
 0.14604207 0.19661092 0.177564   0.08645797 0.02423016 0.01722288
```

Here we have loaded the greyscale output image.

```
(32, 32)
<ipython-input-4-4181bfedb062>:1: FutureWarning: `multichannel` is a deprecated argument name for `hog`. It will be removed in version 1.0. Please use `channel_axis` instead.
  fd, hog_image=hog(resized_img, visualize=True,multichannel=True)
```

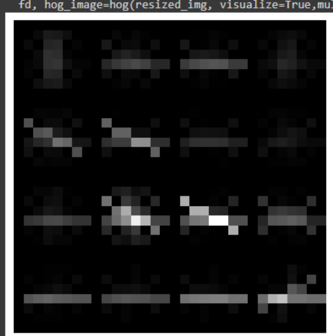

Figure -5 & 6: Feature loading and Greyscale output

6. Inserting Training Dataset

Here the Training dataset was entered for each category from the dataset. The training dataset included 50000 images each category having 5000 images. There were 10 categories.

While importing the images were converted to **greyscale** with the help of **CV2** library file and **CV2.COLOR_BGR2GRAY** function. After that all the values from the dataset were merged in **train_data** variable

```
#For Airplane
data_airplane = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/airplane/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_airplane.append(fd)
#For Automobile
data_automobile = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/automobile/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_automobile.append(fd)
#for bird

data_bird = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/bird/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_bird.append(fd)
```

```
#for bird

data_bird = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/bird/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_bird.append(fd)


#for cat
data_cat= []
for entry in glob.glob("/content/CIFAR-10-images-master/train/cat/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_cat.append(fd)

#for deer
data_deer = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/deer/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_deer.append(fd)

#for dog
data_dog = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/dog/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_dog.append(fd)
```

```
data_frog = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/frog/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_frog.append(fd)

#for horse
data_horse = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/horse/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_horse.append(fd)

#for ship
data_ship = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/ship/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_ship.append(fd)

#for truc
data_truck = []
for entry in glob.glob("/content/CIFAR-10-images-master/train/truck/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32) )
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  data_truck.append(fd)

#combining data
train_data = data_airplane+data_automobile+data_bird+data_cat+data_dog+data_deer+data_frog+data_horse+data_ship+data_truck
print(len(train_data))
```

Figure -7: Loading Training dataset.

7. Labelling training dataset

All the training dataset was labelled.

```
[6] train_lable_list = []
    for i in range (len(train_data)):
      if i < 5000:
        train_lable_list.append('airplane')

      elif 4999< i < 10000:
        train_lable_list.append('automobile')

      elif 9999< i < 15000:
        train_lable_list.append('bird')

      elif 14999< i < 20000:
        train_lable_list.append('cat')

      elif 19999< i < 25000:
        train_lable_list.append('deer')

      elif 24999< i < 30000:
        train_lable_list.append('dog')

      elif 29999< i < 35000:
        train_lable_list.append('frog')

      elif 34999< i < 40000:
        train_lable_list.append('horse')

      elif 39999< i < 45000:
        train_lable_list.append('ship')

      elif 44999< i < 50000:
        train_lable_list.append('truck')

    print(train_lable_list)
    print(len(train_lable_list))

['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', '.
50000
```

Figure -8: Loading Training dataset.

8. Inserting Testing Dataset

Here the Training dataset was entered for each category from the dataset. The training dataset included 10000 images each category having 1000 images. There were 10 categories.

While importing the images were converted to **greyscale** with the help of **CV2** library file and **CV2.COLOR_BGR2GRAY** function. After that all the values from the dataset were merged in **test_data** variable.

```
# Labled Airplane Test Data

test_airplane = []
for entry in glob.glob("/content/CIFAR-10-images-master/test/airplane/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32))
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  test_dict = {'data':fd, 'lable':'airplane'}
  test_airplane.append(test_dict)

# Labled automobile Test Data

test_automobile = []
for entry in glob.glob("/content/CIFAR-10-images-master/test/automobile/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32))
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  test_dict = {'data':fd, 'lable':'automobile'}
  test_automobile.append(test_dict)

  # Labled bird Test Data

test_bird = []
for entry in glob.glob("/content/CIFAR-10-images-master/test/bird/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32))
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  test_dict = {'data':fd, 'lable':'bird'}
  test_bird.append(test_dict)

  # Labled cat Test Data
```

```python
# Labled cat Test Data

test_cat = []
for entry in glob.glob("/content/CIFAR-10-images-master/test/cat/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32))
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  test_dict = {'data':fd, 'lable':'cat'}
  test_cat.append(test_dict)

  # Labled deer Test Data

test_deer = []
for entry in glob.glob("/content/CIFAR-10-images-master/test/deer/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32))
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  test_dict = {'data':fd, 'lable':'deer'}
  test_deer.append(test_dict)

  # Labled dog Test Data

test_dog = []
for entry in glob.glob("/content/CIFAR-10-images-master/test/dog/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32))
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  test_dict = {'data':fd, 'lable':'dog'}
  test_dog.append(test_dict)

  # Labled frog Test Data
```

```python
  # Labled horse Test Data

test_horse = []
for entry in glob.glob("/content/CIFAR-10-images-master/test/horse/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32))
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  test_dict = {'data':fd, 'lable':'horse'}
  test_horse.append(test_dict)

  # Labled ship Test Data

test_ship = []
for entry in glob.glob("/content/CIFAR-10-images-master/test/ship/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32))
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  test_dict = {'data':fd, 'lable':'ship'}
  test_ship.append(test_dict)

  # Labled truck Test Data

test_truck = []
for entry in glob.glob("/content/CIFAR-10-images-master/test/truck/*.jpg"):
  img = np.array(mpimg.imread(entry))
  resized_img = cv2.resize(img,(32,32))
  resized_img_gray = cv2.cvtColor(resized_img, cv2.COLOR_BGR2GRAY)
  fd= hog(resized_img_gray, visualize=False, multichannel=False)
  test_dict = {'data':fd, 'lable':'truck'}
  test_truck.append(test_dict)


#combining test data
test_data = test_airplane+test_automobile+test_bird+test_cat+test_deer+test_dog+test_frog+test_horse+test_ship+test_truck
print(len(test_data))
```

Figure -9: Loading Training dataset.

9. Labels and Features extraction for test data

Labels and Features were extracted separately and were stored in test_features and test_lables array.

After the extraction we can see that the length of the feature array was 1000 meaning all the features were stored successfully

```python
test_features = []
test_lables = []


for i in test_data:
  test_lables.append(i['lable'])
  test_features.append(i['data'])
print(len(test_features))
print(test_lables)
print(test_features[0])
```

```
10000
['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', '
[0.03518072 0.03372132 0.02107034 0.01190998 0.00924711 0.
 0.00489511 0.02218661 0.03085124 0.00659425 0.01782573 0.00830238
 0.18265473 0.09106268 0.01848849 0.00365368 0.00886204 0.00613734
 0.01499521 0.00625197 0.00558707 0.01319591 0.01812663 0.16147782
 0.01863659 0.0076251  0.00449599 0.17871223 0.05424639 0.13838043
 0.07514631 0.13536824 0.1089189  0.07263749 0.18515695 0.12925077
 0.07282535 0.00212073 0.15592819 0.23770208 0.09429442 0.16984984
 0.08298341 0.0355514  0.03066173 0.0176337  0.04342717 0.02799462
 0.06263096 0.09323231 0.15224272 0.23770208 0.23770208 0.0259931
 0.02907589 0.02479118 0.03175479 0.23770208 0.23770208 0.04673465
 0.22665692 0.15021459 0.02881977 0.05088835 0.23770208 0.12509536
 0.07362166 0.23770208 0.07508863 0.04754344 0.04322622 0.05799575
 0.03075538 0.01976512 0.05623829 0.21599937 0.23770208 0.20605943
```

Figure-10: Label and Feature extraction.

10. Prediction and calculation with Euclidean distance (L2)

The Euclidean distance was calculated and the value of hyperparameter K was varied from (1-5).

Power value p was not given, meaning the KNeighborsClassifier() will run with Euclidean distance values.
The Confusion matrix was used to describe the performance of KNN when using Euclidean distance.

```python
x_axis_k_points = []
f1_euclidean = []
accuracies_euclidean = []
conf_matrix_euclidean = []
# Convert train_data and test_labels to NumPy arrays

for k in range(5):
    # KNN CLASSIFIER Train Data
    knn_euclidean = KNeighborsClassifier(n_neighbors=k+1)
    knn_euclidean.fit(train_data, train_lable_list)
    # KNN CLASSIFIER prediction
    pred_labels_euclidean = knn_euclidean.predict(test_features)
    # Accuracy of prediction
    acc_euclidean = knn_euclidean.score(test_features, test_lables)
    accuracies_euclidean.append(acc_euclidean)
    # Confusion Matrix of prediction
    conf_matrix_euclidean.append(metrics.confusion_matrix(test_lables, pred_labels_euclidean))
    # F1 Score of prediction
    f1_euclidean.append(metrics.f1_score(test_lables, pred_labels_euclidean, average='micro'))
    x_axis_k_points.append(k+1)
    print(f"K = {k}")
    print("Predicted Labels:", pred_labels_euclidean)
    print("Actual Labels:", test_lables)
    print("------------------------------------")


# Print results
print("F1 Scores:", f1_euclidean)
print("Accuracies:", accuracies_euclidean)
print("Confusion Matrices:", conf_matrix_euclidean)
```

Figure-10: Runnng KNN with Euclidean distance

11. Accuracy Calculation for Euclidian Distance

As we can see from the result shown below, for each value of K we got a prediction. We have used F1 scoring metric for evaluation and finding the value of accuracy for this classification model.

$$F1\ Score = 2\ \times \frac{Precision\ \times Recall}{Precision + Recall}$$

The above formula was used to calculate the accuracy value.

Also, predicted Result was shown Below

F1 Scores: 0.419999, 0.3953, 0.295, 0.4324, 0.4372

```
K = 0
Predicted Labels: ['ship' 'ship' 'dog' ... 'truck' 'frog' 'cat']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplan
------------------------------------
K = 1
Predicted Labels: ['cat' 'airplane' 'dog' ... 'ship' 'frog' 'cat']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplan
------------------------------------
K = 2
Predicted Labels: ['bird' 'airplane' 'frog' ... 'truck' 'frog' 'cat']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplan
------------------------------------
K = 3
Predicted Labels: ['bird' 'ship' 'frog' ... 'truck' 'frog' 'cat']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplan
------------------------------------
K = 4
Predicted Labels: ['bird' 'ship' 'frog' ... 'truck' 'frog' 'cat']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplan
------------------------------------
F1 Scores: [0.41469999999999996, 0.3953, 0.4295, 0.4324, 0.4372]
Accuracies: [0.4147, 0.3953, 0.4295, 0.4324, 0.4372]
```

Figure-11: Accuracy Calculation and Prediction

12. Prediction & calculation with Manhattan distance (L1)

The manhattan distance was calculated and the value of hyperparameter K was varied from (1-5).

The power parameter was set with p=1 meaning the KNeighborsClassifier() will run with manhattan distance values. The Confusion matrix was used to describe the performance of KNN when using Euclidean distance.

```
# LIST OF METRICES
f1_manhattan = []
accuracies_manhattan = []
conf_matrix_manhattan = []
for k in range(5):
  # KNN CLASSIFIER Train Data
  knn_manhattan = KNeighborsClassifier(n_neighbors=k+1, p=1)
  knn_manhattan.fit(train_data,train_lable_list)
  # KNN CLASSIFIER prediction
  pred_lables_manhattan = knn_manhattan.predict(test_features)
  # Accuracy of prediction
  acc_manhattan = knn_manhattan.score(test_features, test_lables)
  accuracies_manhattan.append(acc_manhattan)
  # Confusion Matrix of prediction
  conf_matrix_manhattan.append(metrics.confusion_matrix(test_lables, pred_lables_manhattan))
  # F1 Score of prediction
  f1_manhattan.append(metrics.f1_score (test_lables, pred_lables_manhattan, average='micro'))

  print(f"K = {k}")
  print("Predicted Labels:", pred_lables_manhattan)
  print("Actual Labels:", test_lables)
  print("------------------------------------")

print("F1 Scores:", f1_manhattan)
print("Accuracies:", accuracies_manhattan)
print("Confusion Matrices:", conf_matrix_manhattan)
```

Figure-12: Runnng KNN with Manhattan distance

13. Accuracy Calculation for Manhattan Distance

As we can see from the result shown below, for each value of K we got a prediction.
We have used F1 scoring metric for evaluation and finding the value of accuracy for this classification model.

$$F1\ Score = 2 \times \frac{Precision\ \times Recall}{Precision + Recall}$$

The above formula was used to calculate the accuracy value.

The Prediction Result was also shown below

Accuracy Scores: 0.4293, 0.4075, 0.4432, 0.4507, 0.4555

```
K = 0
Predicted Labels: ['ship' 'airplane' 'frog' ... 'truck' 'truck' 'cat']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', '
------------------------------------
K = 1
Predicted Labels: ['airplane' 'airplane' 'frog' ... 'ship' 'truck' 'cat']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', '
------------------------------------
K = 2
Predicted Labels: ['airplane' 'ship' 'frog' ... 'truck' 'truck' 'cat']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', '
------------------------------------
K = 3
Predicted Labels: ['ship' 'airplane' 'frog' ... 'truck' 'truck' 'dog']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', '
------------------------------------
K = 4
Predicted Labels: ['airplane' 'airplane' 'frog' ... 'truck' 'truck' 'cat']
Actual Labels: ['airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', 'airplane', '
------------------------------------
F1 Scores: [0.4293, 0.4075, 0.4432, 0.4507, 0.4555]
Accuracies: [0.4293, 0.4075, 0.4432, 0.4507, 0.4555]
```

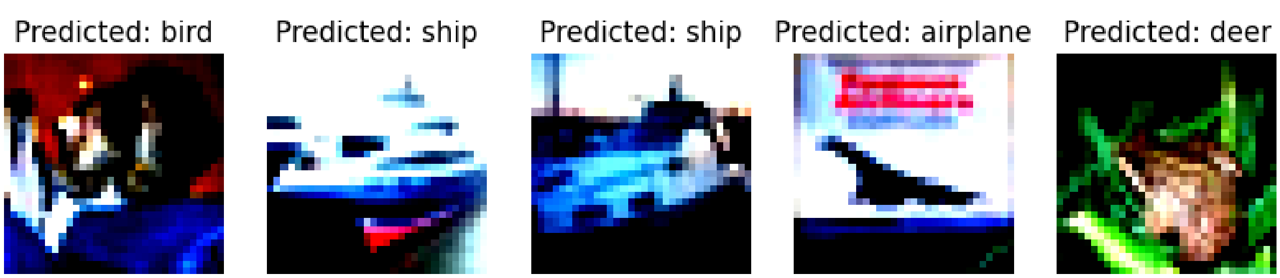

Figure-13: Accuracy Calculation & Prediction

14. Accuracy Calculation for all 5 images

From the image below we can see accuracy value with accordance to confusion matrix was shown for all 5 images

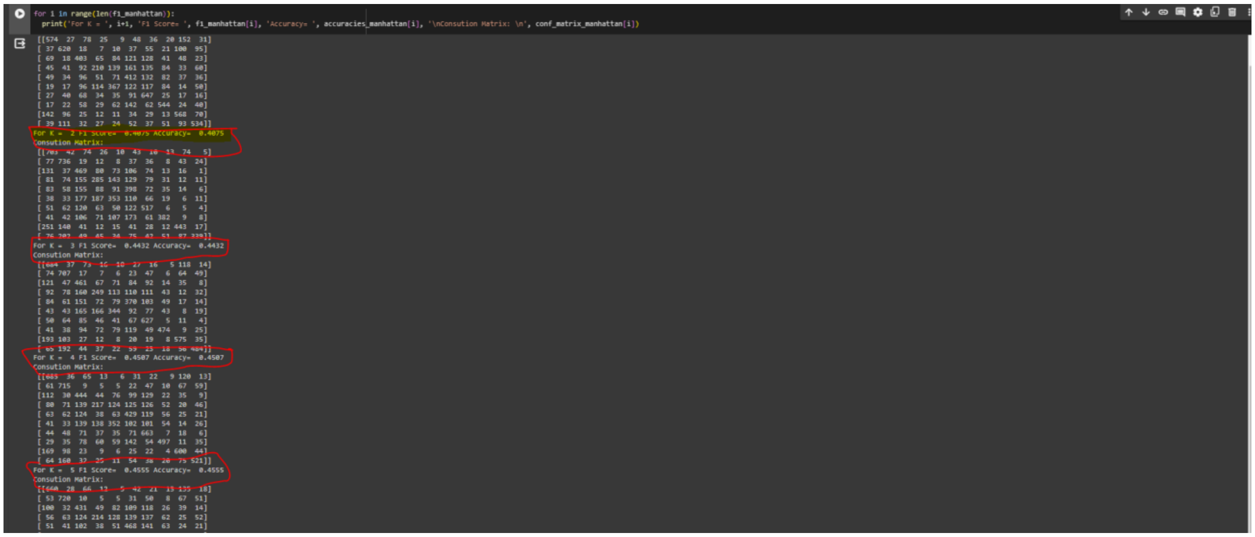

Figure-14: Accuracy Calculation with confusion matrix

15. F1 Score Comparison for Euclidian and Manhattan

We know that KNN is a **"Lazy Learning Algorithm",** we can verify this from the below image. At first the accuracy was very low and by time KNN started to improve its accuracy hence it gradually improves and learns.

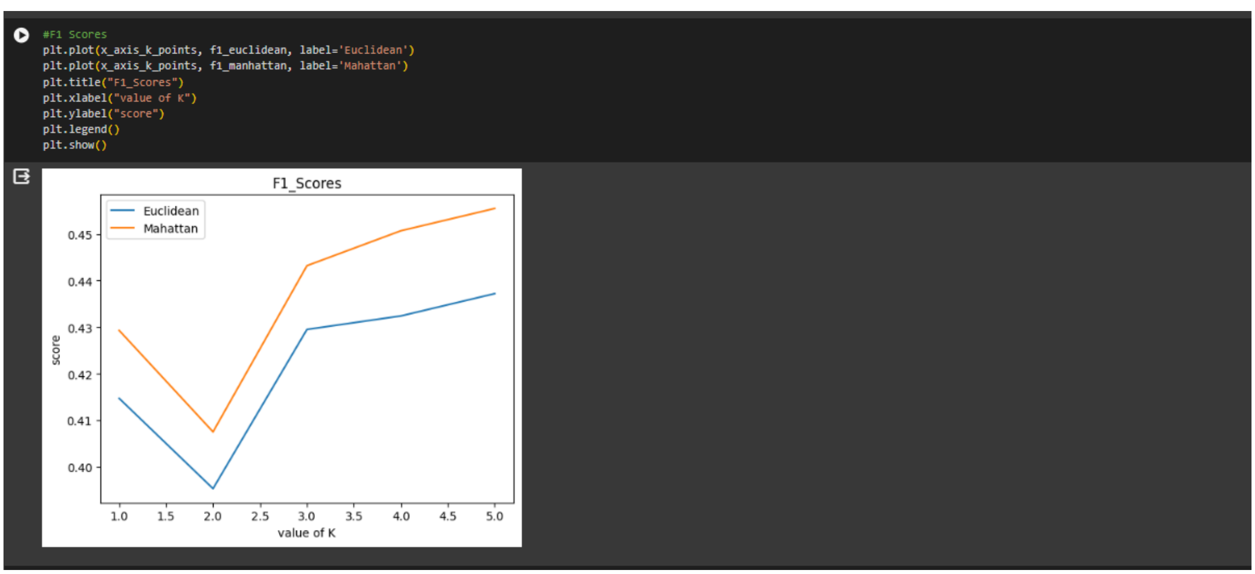

Figure-15: F1 Score Comparison

16. Accuracy Comparison for Euclidian and Manhattan

Like F1 Score Comparison, same thing can be seen here. KNN improves its accuracy over time by leering. But this learning rate is very low.

```
[ ] #Accuracies
    plt.plot(x_axis_k_points, accuracies_euclidean, label='Euclidean')
    plt.plot(x_axis_k_points, accuracies_manhattan, label='Mahattan')
    plt.title("Accuracies")
    plt.xlabel("value of K")
    plt.ylabel("Accuracy")
    plt.legend()
    plt.show()
```

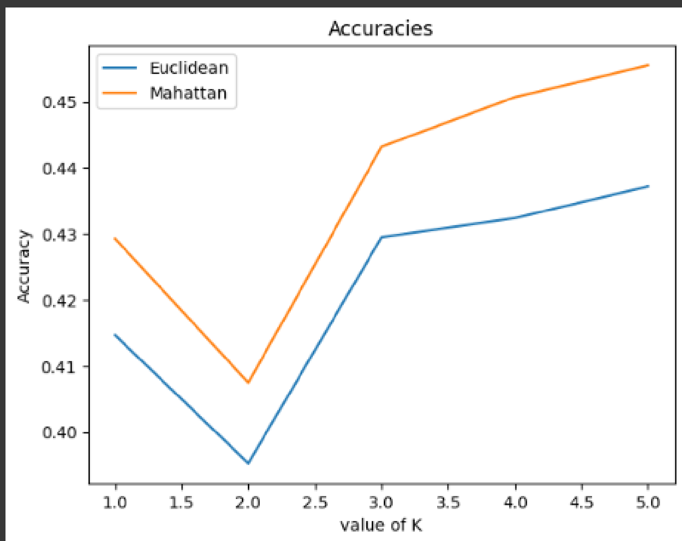

Figure-16: Accuracy Comparison