Date out : May 15, 2020 Due on: May 25, 2020

ENIN 880CA – Spring/Summer 2020

Exercise #2 – Deep Learning for Computer Vision – Image Classification using K-NN.

Students should read chapters 1 to 7 (all inclusive) from the pdf file sent to the students' list via email.

Students should watch videos on chapters 1 to 7, shared via Dropbox.

Students should learn about the code explained in chapter 7 inside out, and then copy and run the code (i.e., image classification using K-NN method).

Students should follow the following structure for folders and files they create:

- Root folder: It should include the file: "knn.py" and the folder: "pyimagesearch".
- Folder "pyimagesearch should include the file: "-init-.py" and two folders: "datasets" and "preprocessing".
- The folder "datasets" should include files: "-init-.py" and "simpledatasetloader.py".
- The folder "preprocessing" should include two files: "-init-.py" and "simplepreprocessor.py"

Using an image dataset that includes images of "cats", "dogs", and "pandas", insert an unknown image (e.g. an image of a dog for instance) into the image dataset and then make a classification. Repeat this for random pictures of cats and pandas. Which class will give you the highest rate of success, and why?

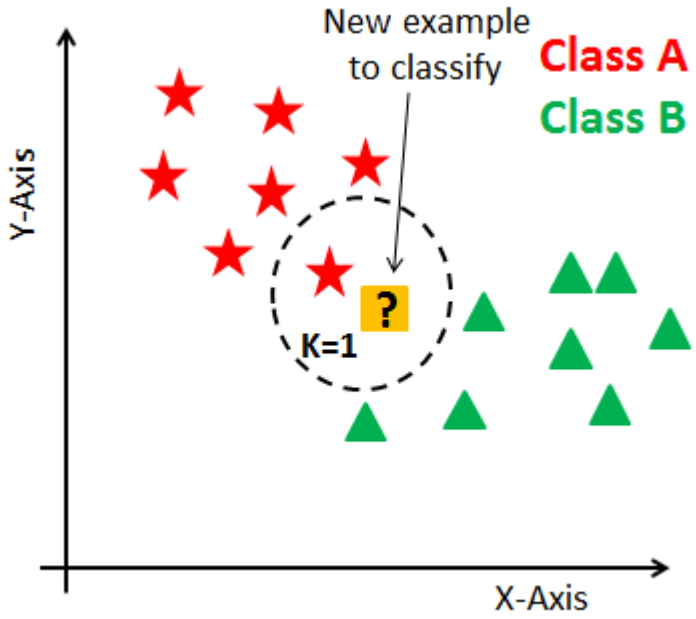
NOTE: the purpose for this exercise is to learn the basics of the K-NN for classification. K-NN works fast, but it does not actually "learn" anything (i.e., an open-loop classification algorithm with no back propagation of parameters and/or feedback). However, it is the simplest "classification" algorithm one can think of and start with. People usually use K-NN as a baseline.

A. Theory: K NN Classification [1]

K Nearest Neighbor (KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN is used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition. KNN algorithm is widely used for both classification and regression problems. KNN algorithm works based on feature similarity approach.

How does the KNN algorithm work?

In KNN, K is the number of nearest neighbors. The number of neighbors is the core decision factor. K is generally an odd number if the number of classes is 2. When K=1, the algorithm is known as the nearest neighbor algorithm. This is the simplest case. Suppose P1 is the point, for which label needs to predict. First, you find the one closest point to P1 and then the label of the nearest point assigned to P1.

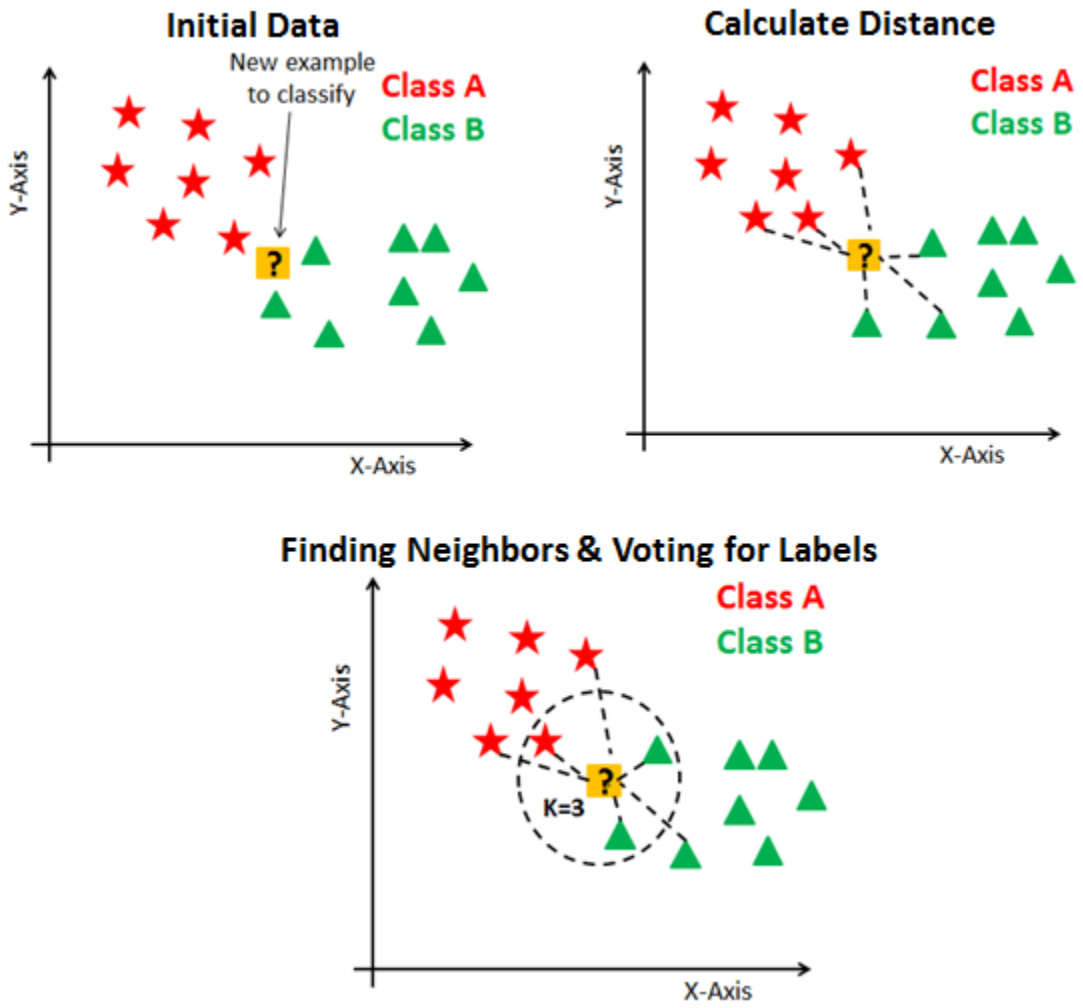


Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance. KNN has the following basic steps:

1. Calculate distance

2. Find closest neighbors

3. Vote for labels



B. Implementing k-NN Classification [2]

The goal of this section is to train a k-NN classifier on the raw pixel intensities of the Animals dataset and use it to classify unknown animal images. We'll be using following steps to train, test, and evaluate classifiers:

• Step #1 – Gather Our Dataset

We use two datasets:

Dataset for Training and Testing: Animals datasets [3]

We are using Animals dataset. The Animals datasets consists of 3,000 images with 1,000 images per dog, cat, and panda class, respectively. A sample of this dataset is represented in figure 1.

In this application we have borrowed 100 panda images for evaluation. Therefore we have used only 900 images per dog, cat, and panda class to maintain the balance between data classes.

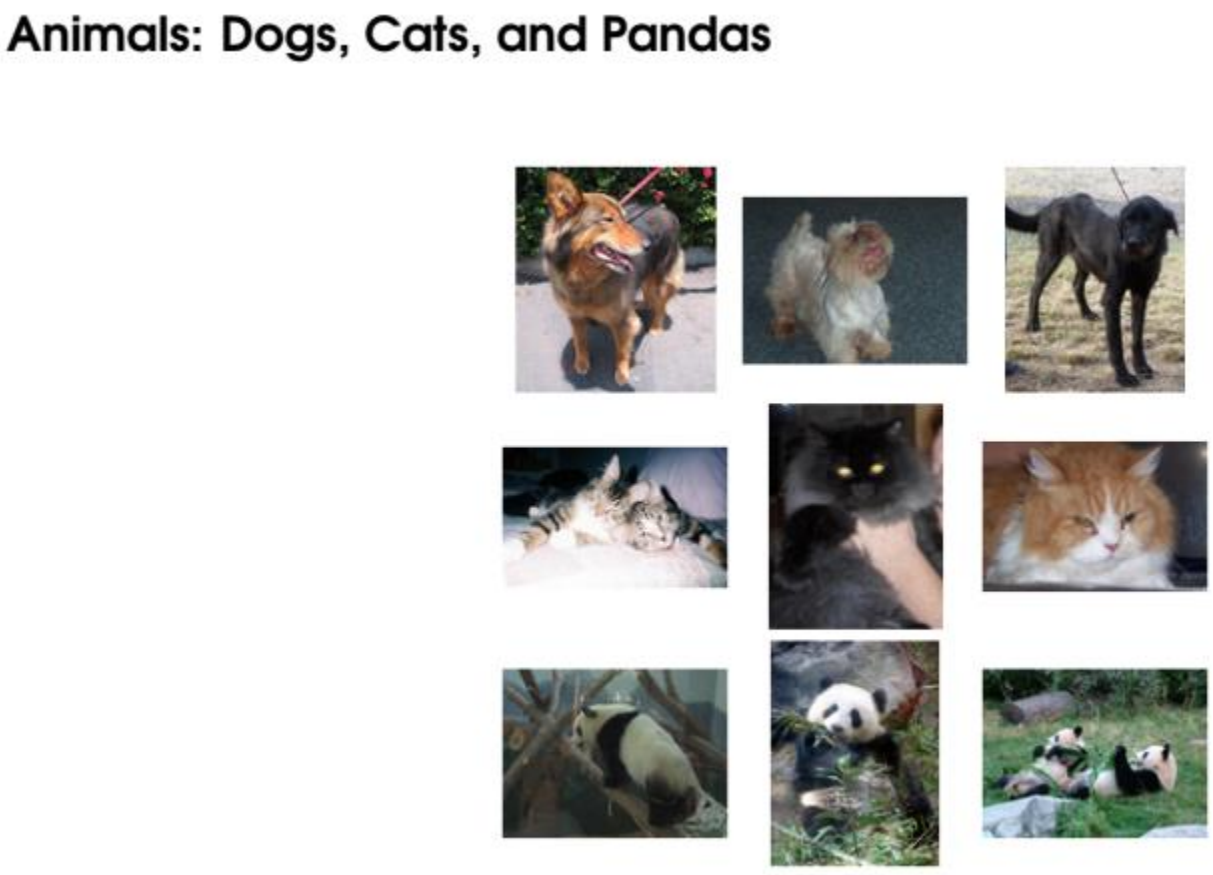


Figure 1: Animals Dataset

Dataset for Evaluation: Cifar-10 datasets [4] & Animals datasets

The evaluation datasets consists of 300 images with 100 images per dog, cat, and panda class. Cat and dog images are from Cifar-10 dataset (figure 2) while panda images have been cut from Animals dataset.

Note: Each image has been used in ONLY one of training, testing or evaluation datasets.

Figure 2: Cifar-10 Dataset

We will preprocess each image by resizing it to 32×32 pixels. Taking into account the three RGB channels, the resized image dimensions imply that each image in the dataset is represented by 32×32×3 = 3,072 integers.

Main Function:

```
image = cv2.imread(image_path)

label = image_path.split(os.path.sep)[-2]

cv2.resize(image, (self.width, self.height), interpolation=self.interpolation)

# Reshape from (3000, 32, 32, 3) to (3000, 32*32*3=3072)

data = data.reshape((data.shape[0], 3072))
```

• Step #2 – Split the Dataset

For this simple example, we'll be using three splits of the data for training, testing and evaluation. Training and testing data are obtained from splitting Animals dataset:

```
Main Function:

# Split data into training (75%) and testing (25%) data

(train_x, test_x, train_y, test_y) = train_test_split(data, labels,
test_size=0.25, random_state=42)
```

Evaluation data is loaded from separate directory (similar to step 1) and therefore does not need to be split.

• Step #3 – Train the Classifier for Different Parameters (n_neighbors, weights, algorithm)

Our k-NN classifier will be trained on the raw pixel intensities of the images in the training set.

To find most appropriate parameters, we would train 120 models using different parameters and compare their precision. Our parameters are as follows:

- **Weights**: {'uniform', 'distance'}
- **Algorithm**: {'auto', 'ball_tree', 'kd_tree', 'brute'}
- **n_neighbors**: {1,2,3,4,5,6,7,8,9,10,12,14,16,18,20}

Therefore, we would test 2 x 4 x 15 = 120 combinations of parameters, and compare their average precision and then decide which parameter choice would have best results.

```
Main Function:
model[m] = KNeighborsClassifier(n_neighbors= neighbors[n], weights= weights[w],
algorithm= algorithm[a])
model.fit(train_x, train_y)
```

• Step #4 – Testing the Classifier for Different Parameters (n_neighbors, weights, algorithm)

After training m models, we can evaluate performance on the test set and save the results for comparison

```
Main Function:
result[m] =
classification_report(test_y,model[m].predict(test_x),target_names=le.classes_)
```

• Step #5 – Evaluate Tuned Classifier using Unknown Data

Using the results from previous step, we can decide which values for parameters (n_neighbors, weights, algorithm) would perform best. Then, we evaluate the tuned model using unknown data (from Cifar-10 dataset).

```
Main Function:
print(classification_report(label_eval,model.predict(data_eval),target_names=le.classes_))
```

C. Performance Results for k-NN Classifier

C.1) k-NN Classifier Average Performance for Test Data [Varied Parameters: n_neighbors, weights, algorithm]

The average precision of k-NN Classifier for for varied parameters (n_neighbors, weights, algorithm) is shown in figure 3.

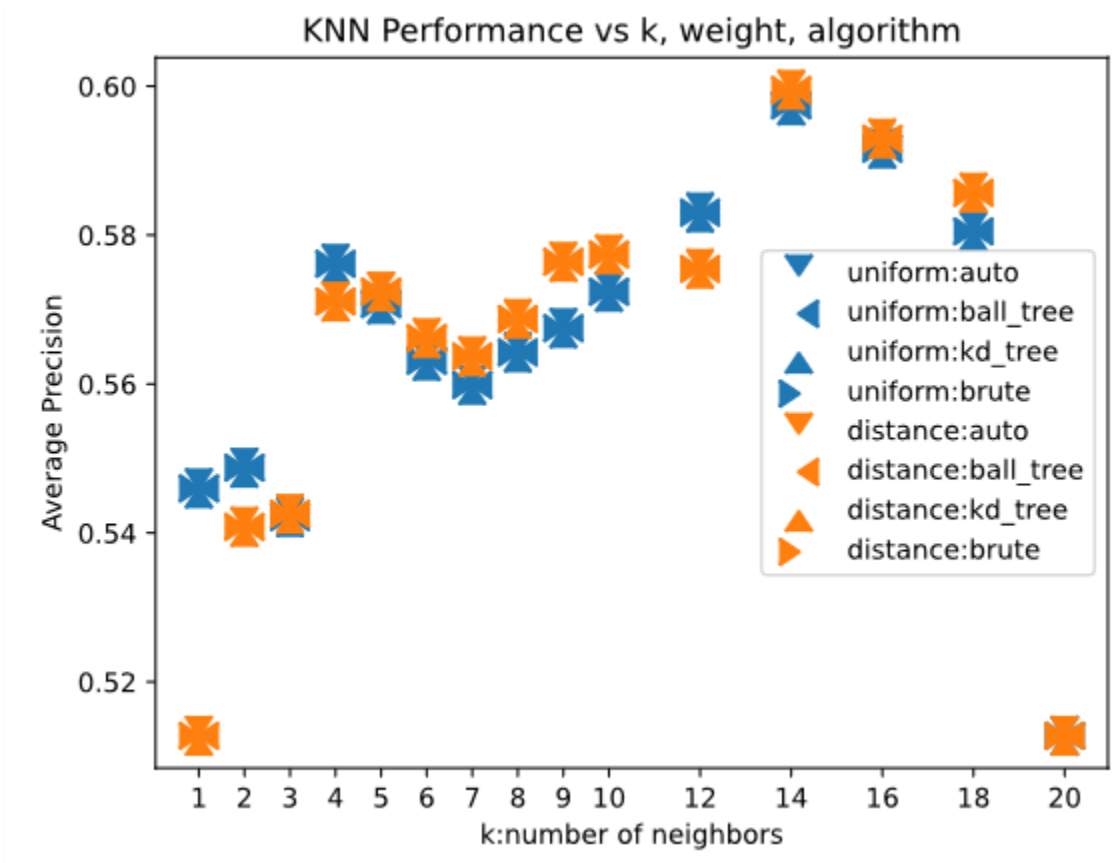


Figure 3: k-NN Classifier Average Performance for Test Data

As illustrated in figure 3:

- Average precision is between 54 to 58 percent;
- For most n_neighbors values, the average precision of “distance” weight is slightly higher than “uniform” weight;
- For all n_neighbors values, the average precision is equal for all algorithms;
- Among all algorithms, “brute” was fastest;
- Considering n_neighbors values, the average precision is highest for n_neighbors = 14.

C.2) K-NN Classifier Class-wise Performance for Test Data [Varied Parameters: n_neighbors, weights, class]

In addition to the average precision of kNN classifier, we have also examined k-NN Classifier precision for our three classes (Cats, Dogs, Panda). The result is shown in figure 4.

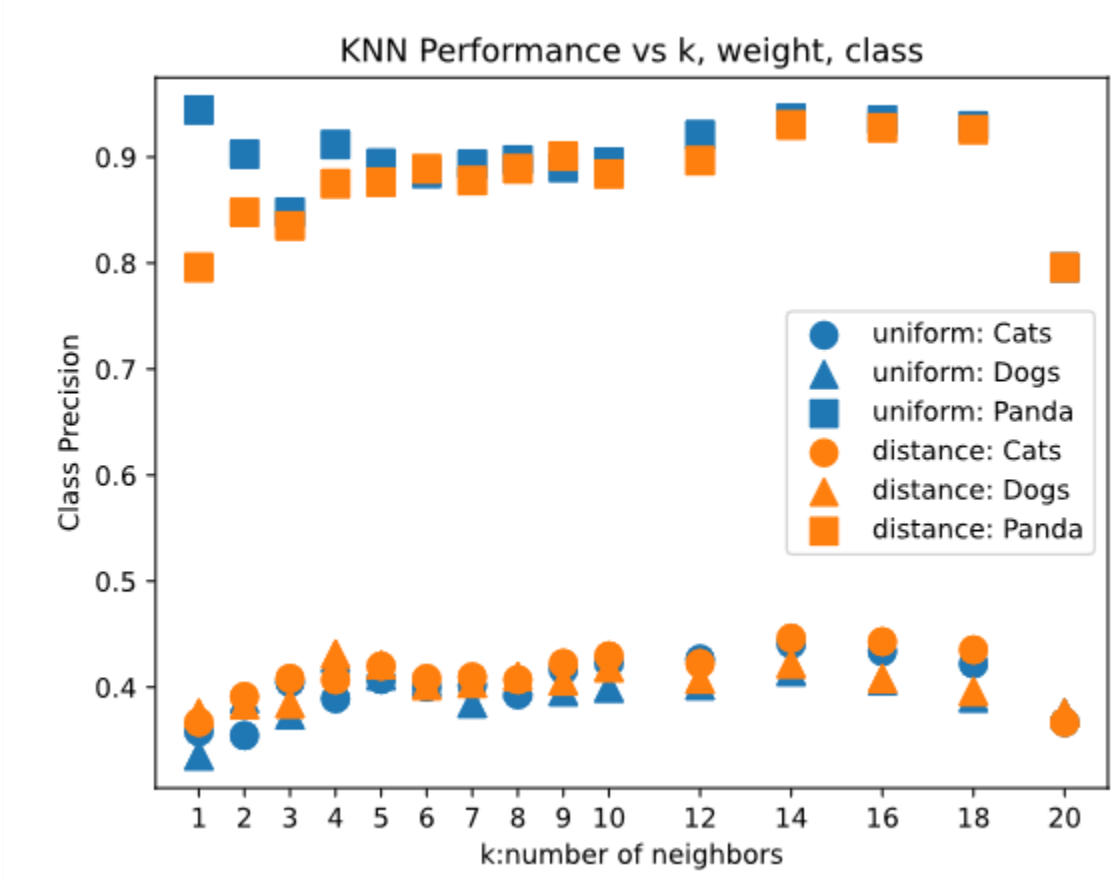


Figure 4: K-NN Classifier Class-wise Performance for Test Data

As illustrated in figure 4:

- Average precision is about 90% for Panda and 40% for Cats and Dogs. This significant difference can be attributed to the fact that dogs and cats can have very similar shades of fur coats and the color of their coats cannot be used to discriminate between them.

- For Panda class, the average precision of “uniform” weight is slightly higher than “distance” weight. However for Cats and Dogs, “distance” weight has resulted higher precision.
- This significant difference can be attributed to the fact that dogs and cats can have very similar shades of fur coats and the color of their coats cannot be used to discriminate between them.

C.3) k-NN Classifier Class-wise and Average Performance for Evaluation Data [Varied Parameters: n_neighbors, class]

Finally, we have evaluated k-NN Classifier using evaluation data. This data has not been used for training or testing. The result is shown in figure 5.

According to test results, following values were used for our kNN classifier:

- n_neighbors = [2,4,6,8,10,12,14,16,18,20];
- weights = “distance”
- algorithm = “brute”

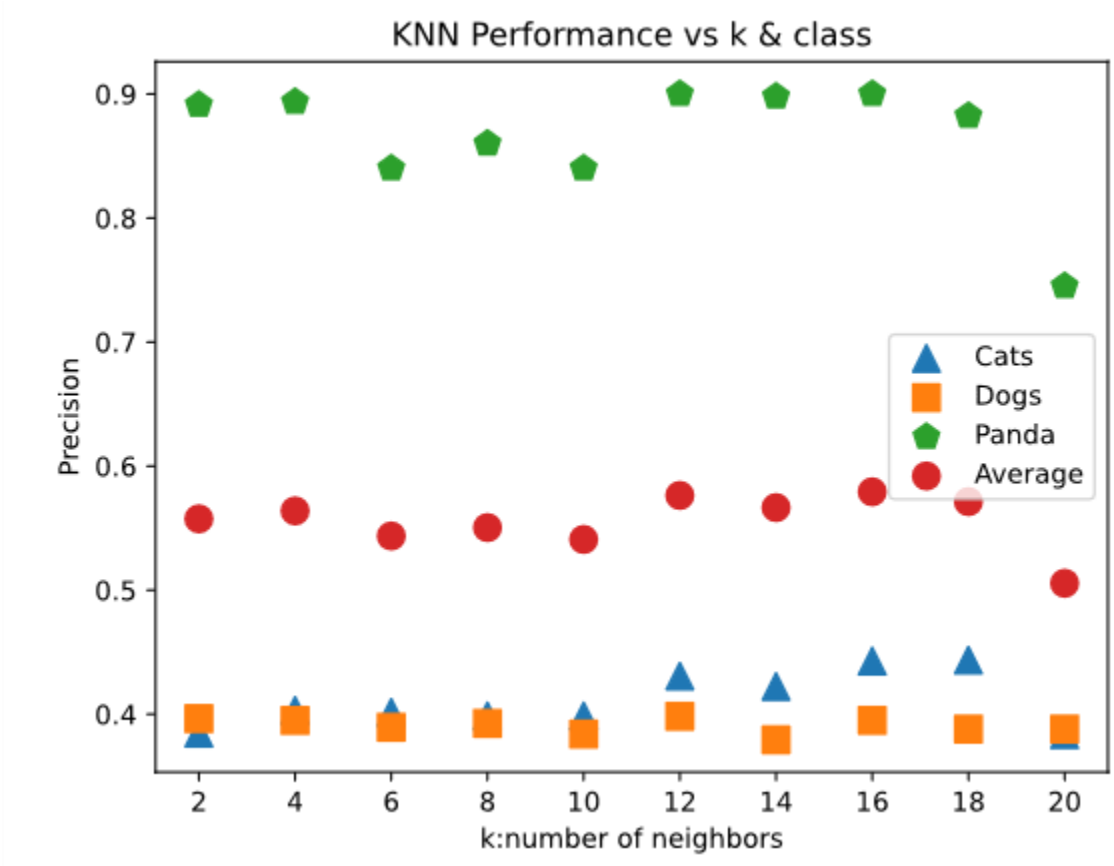


Figure 5: k-NN Classifier Class-wise and Average Performance for Evaluation Data

As illustrated in figure 5:

- Overall average precision is 56%
- Average class precision is 41% for Cats, 39% for Dogs, and 87% for Panda.

- Considering n_neighbors values, the average precision is highest for n_neighbors = 12, and then for n_neighbors = 2.

C.4) Discussion

Simplicity is the main advantage of k-NN classification. However, it is not capable of precise classification. As illustrated in figures 4 and 5, average class precision is more that 85% for Panda and 40% for Cats and Dogs.

The reason behind this low precision for cats and dogs is their coat color. The k-NN classifier used two features for classification: the "fluffiness" of the animal's coat and the "lightness" of their coat.

Since dogs and cats can have very similar shades of fur coats, the classifier cannot distinguish them. On the other hand, panda are only in black and white and so easily distinguishable by the classifier.

References

[1] https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn
[2] book 'Deep Learning for Computer Vision - Starter Bundle' by Adrian Rosebrock
[3] Dataset Animals available on https://github.com/Abhs9/DL4CVStarterBundle
[4] Dataset Cifar-10 available on https://pjreddie.com/projects/cifar-10-dataset-mirror/