

I SRN declare that I have completed this assignment completely and entirely on my own, without any consultation with others.  I understand that any breach of the UAB Academic Honor Code may result in severe penalties.

Srushti R Nayak

SP2024 CS Deep Learning

Hw1: KNN

**KNN algorithm for Image classification**

1. **Background and Method Introduction:**

KNN stands for K-Nearest Neighbours. It is a non-parametric algorithm and it is used for classification and regression problems. To predict class label of a data point it uses its nearest k neighbours in the feature space.

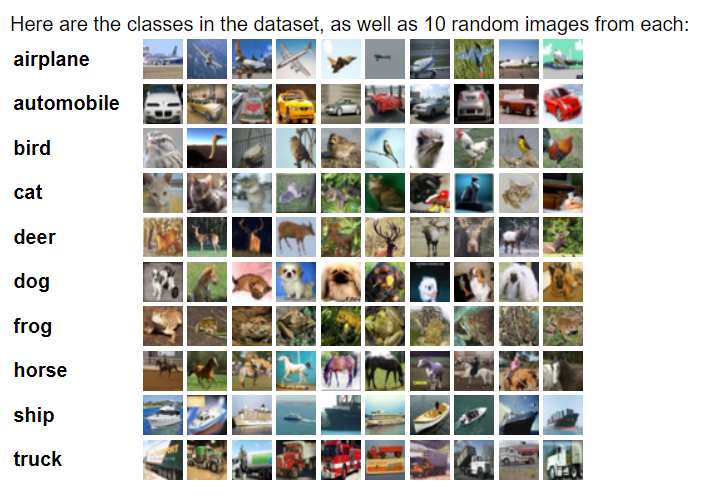
KNN can be used to perform image classification. By comparing pixel values, we can compare similarity of images. In this report, CIFAR-10 dataset is used to perform image classification. This report presents the implementation of the KNN algorithm using both a library-based function (scikit-learn) and a self-coded version.

1. **Dataset and Tasks Description**:

**2.1. CIFAR10 dataset:**

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.



**Fig1: Image classes in CIFAR10 database**

**Image from:** [**https://www.cs.toronto.edu/~kriz/cifar.html**](https://www.cs.toronto.edu/~kriz/cifar.html)

**2.2. Tasks Description:**

In this report, KNN is performed on CIFAR10 dataset.

At first, inbuilt KNN algorithm from scikit-learn is implemented on entire dataset.

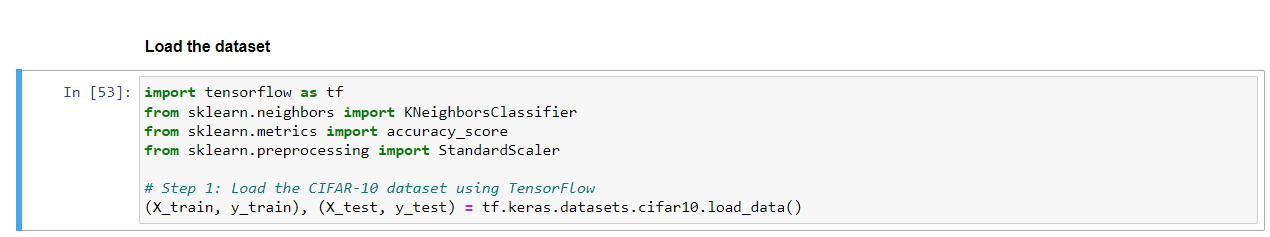
Then self-implemented KNN algorithm is applied on sample from CIFAR10 dataset.

This self-implemented KNN algorithm is improved by hyperparameter tuning and cross validation.

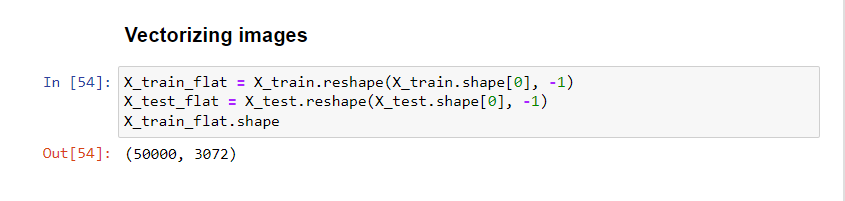
1. **Algorithms Used**:

**3.0 Loading dataset and preparing data:**

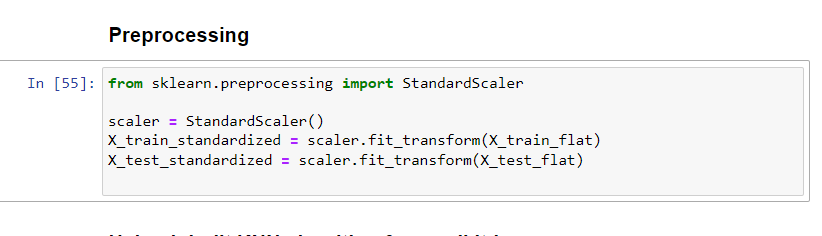
- Tensorflow library is used to load the dataset.



* Converting images into vector form that is suitable input to KNN algorithm

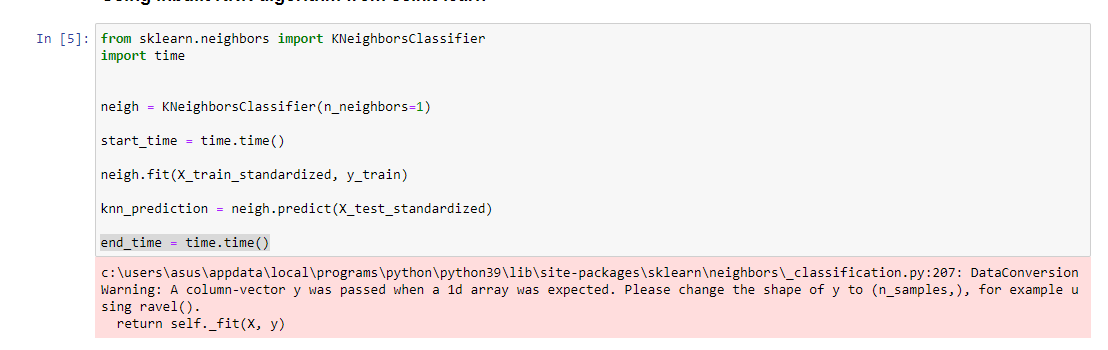


* Applying standardization on features

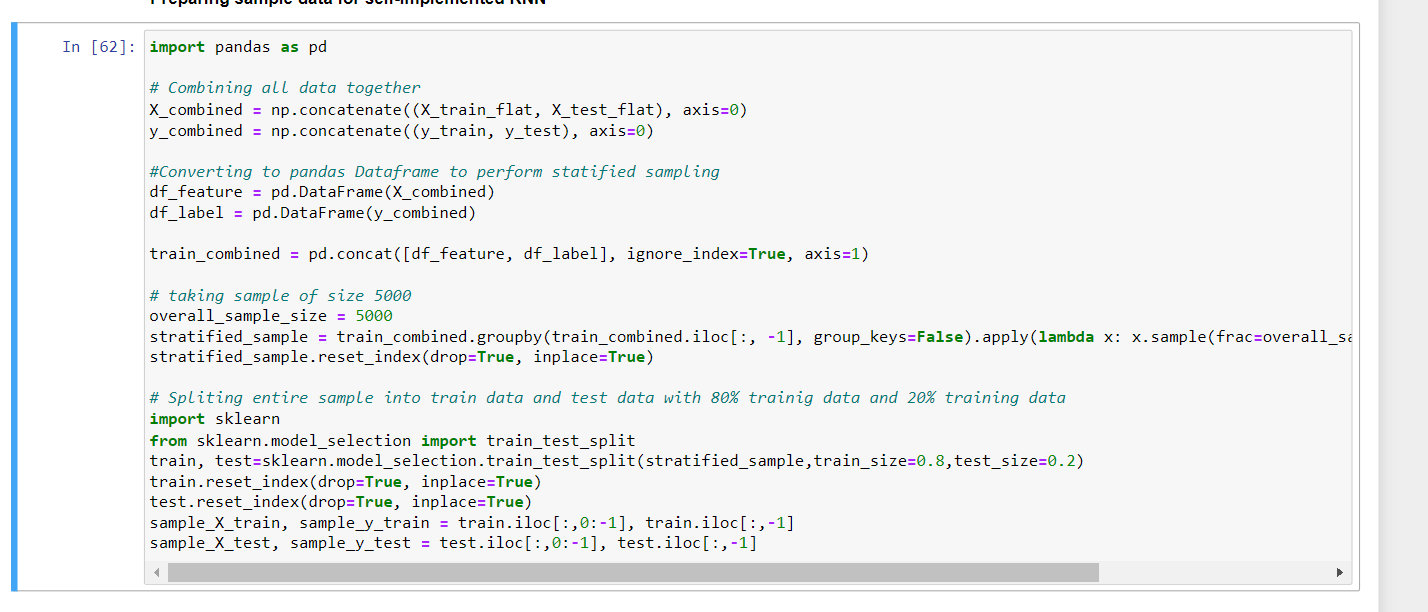


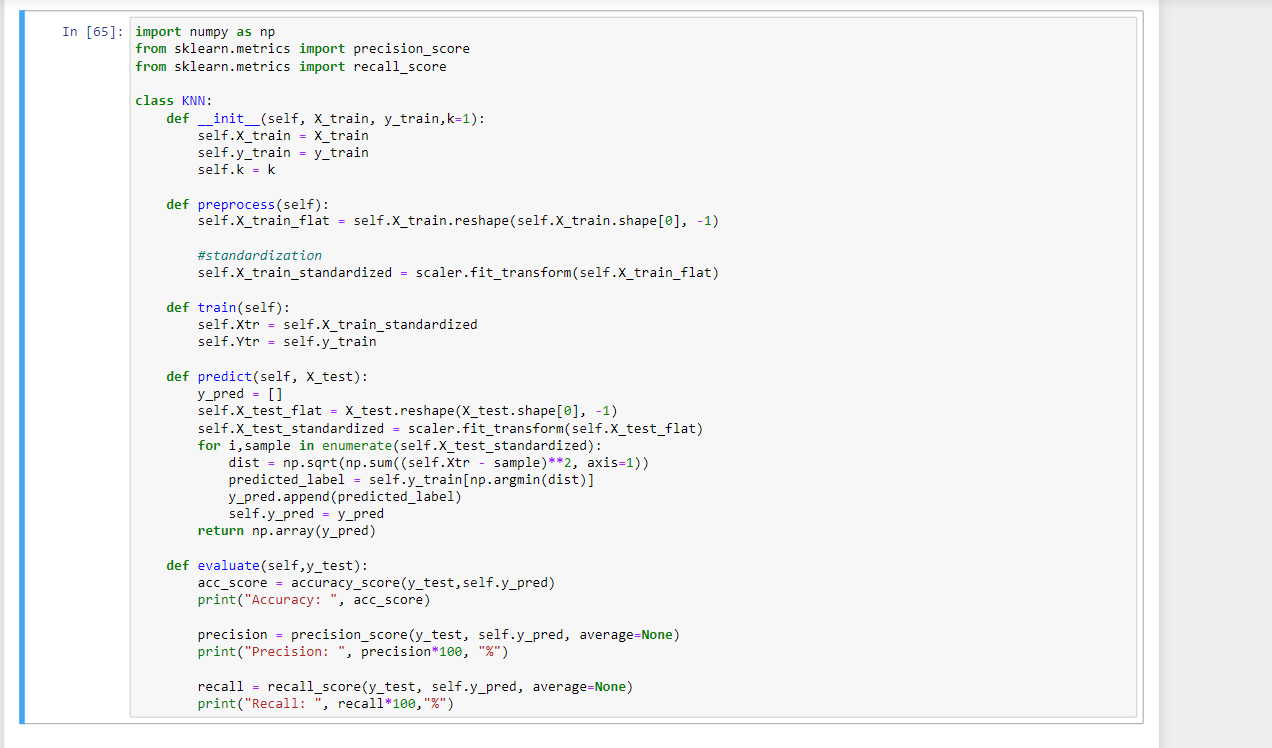
**3.1 Library based KNN:**

* Making prediction using inbuilt KNN algorithm from scikit-learn

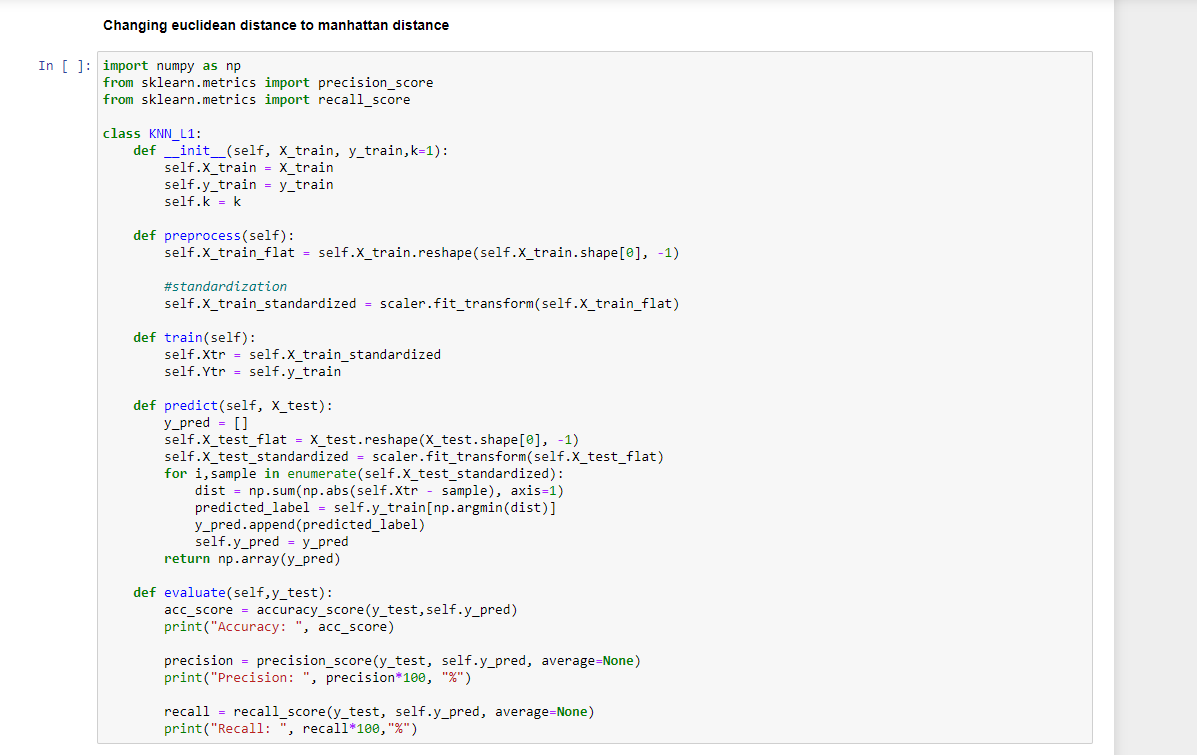
****

**3.2 Self-implemented KNN**

* Sampling data from large CIFAR10 dataset****
* Self-Implemented KNN:

****

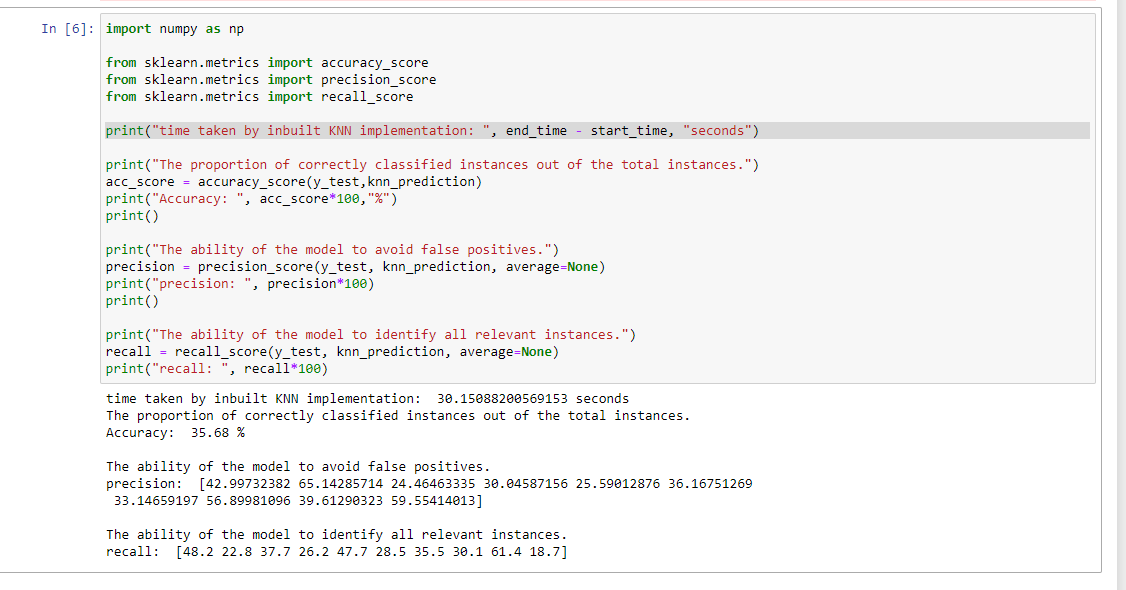
* Self-implemented KNN with Manhattan distance:



1. **Results**

**4.1 Library based KNN**

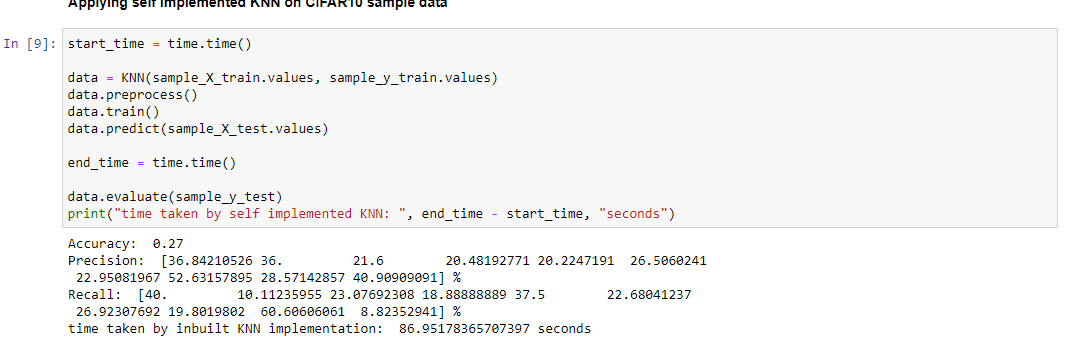
* Evaluating model efficiency
* Accuracy is 35.68%
* Time taken : 30.150 seconds

****

**4.2 Self-implemented KNN**

Accuracy is 25.4%

Time taken: 86.951 seconds

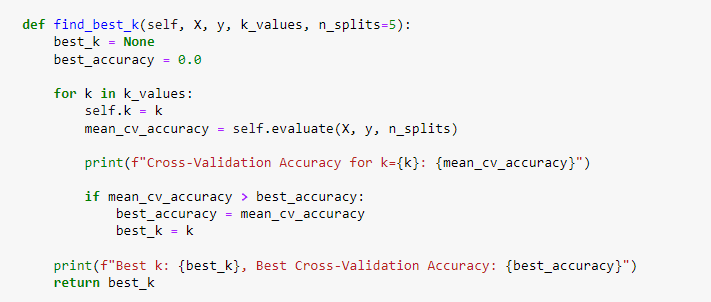
****

**4.3 Self-implemented KNN with Manhattan distance**

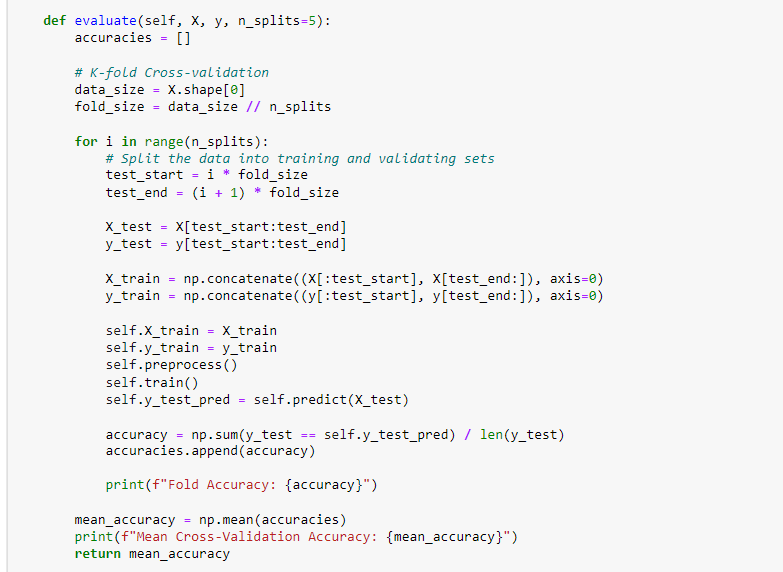
Accuracy is %

1. **Methods of improvement**

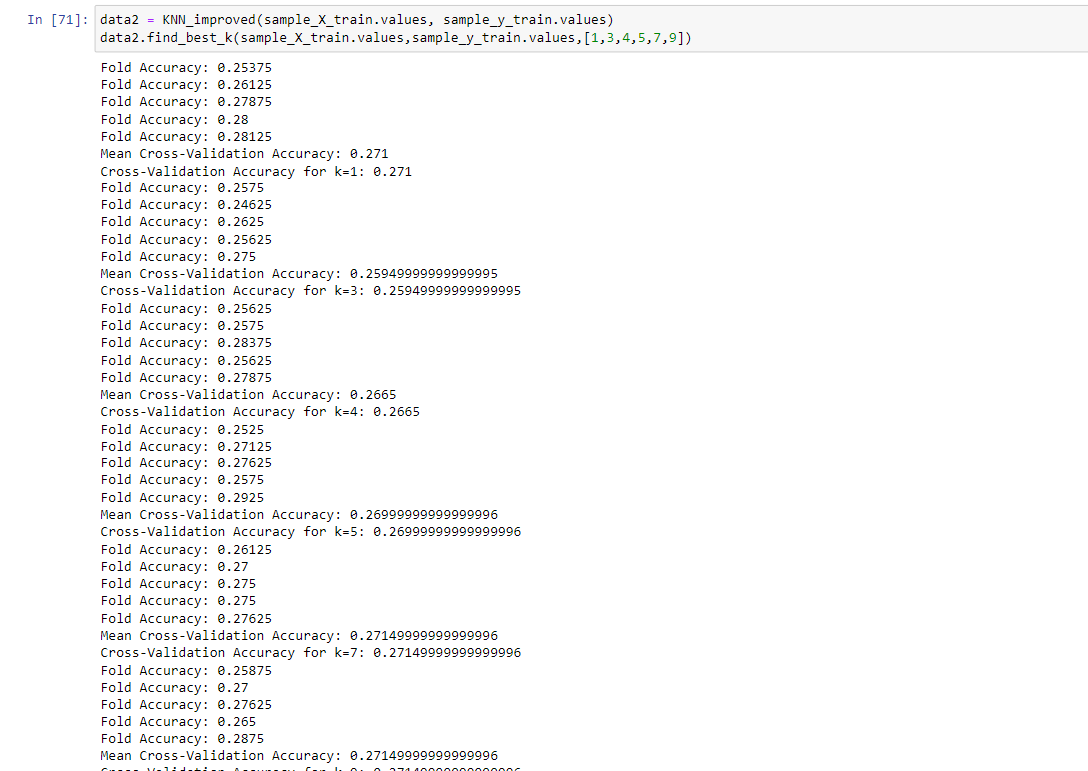
* Performed hyperparameter tuning(by finding best value of k) and cross validation to improve efficiency.
* Finding best value of k

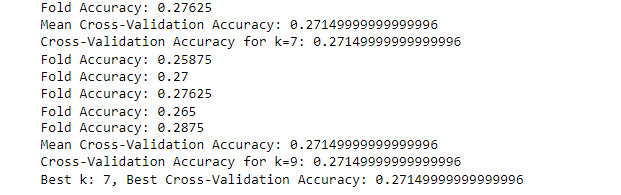


* Using cross validation to evaluate different values of k

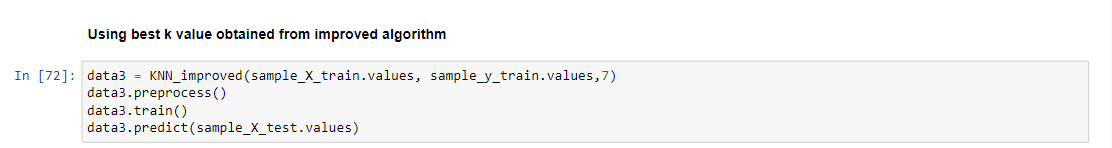
****

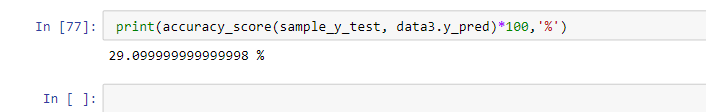
**Improved efficiency after hyperparameter tuning:**

****

****

* Performance on test data



****

* **Hyperparameter Tuning:**

Hyperparameters, distinguished by the 'hyper\_' prefix, serve as top-level controls influencing the learning process and determining the values of model parameters acquired by a learning algorithm. These parameters are pivotal in shaping the learning dynamics and the subsequent model parameter values that evolve during the process.

The improved KNN includes hyperparameter tuning to find the optimal value of k for enhanced accuracy.

* **Standardization:**

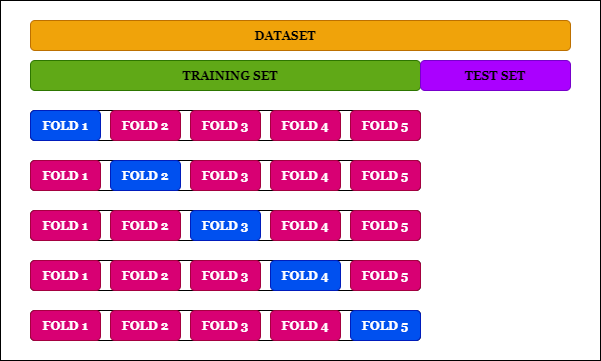
After standardization, mean value becomes close to 0 and standard deviation becomes close to 1.

Standardizing features for improved convergence and model stability.

* **Cross-Validation:**

During k fold cross validations, training data is divided into k parts and iteratively each part works as test data and rest of the k-1 parts behaves as test data and this accuracy is calculated.

Employing cross-validation to assess the model's performance robustly.



**Fig 2: cross validation**

**Image from:** [**https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538**](https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538)

* **Difference in accuracy:**

Accuracy is increased from 25.4% to 29.09% after using best value of k.

* **Difference in speed:**

Speed is reduced due to additional calculation involved in cross validation.

1. **Conclusion:**

To summarize, this report delved into K-Nearest Neighbors (KNN) for CIFAR-10 image classification. Scikit-learn achieved 35.68% accuracy, while the self-implemented KNN reached 25.4%. Hyperparameter tuning improved the self-implemented KNN's accuracy to 29.09%, albeit with a speed trade-off due to cross-validation. This study contributes insights into KNN's strengths, limitations, and the significance of parameter tuning in image classification.

**References:**

[**https://www.cs.toronto.edu/~kriz/cifar.html**](https://www.cs.toronto.edu/~kriz/cifar.html)

[**https://scikit-learn.org/stable/**](https://scikit-learn.org/stable/)

[**https://www.tensorflow.org/api\_docs/python/tf/keras/datasets/cifar10/load\_data**](https://www.tensorflow.org/api_docs/python/tf/keras/datasets/cifar10/load_data)

**Lecture slides**