Week 4 Homework 2: Classification on Colab using MNIST dataset

GitHub link: https://github.com/kathyshe/Machine-Learning/tree/main/Supervised-Learning/Classification%20on%20Colab%20using%20MNIST%20dataset

Introduction

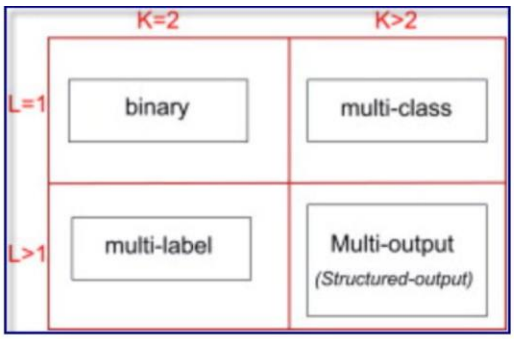
The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems. This dataset is one of the most common datasets used for image classification and accessible from many different sources.

MNIST dataset, which is a set of 70,000 small images of digits handwritten by high school students and employees of the US Census Bureau. Each image is labeled with the digit it represents. This set is called the "hello world" of Machine Learning: Whenever people come up with a new classification algorithm they are curious to see how it will perform on MNIST.

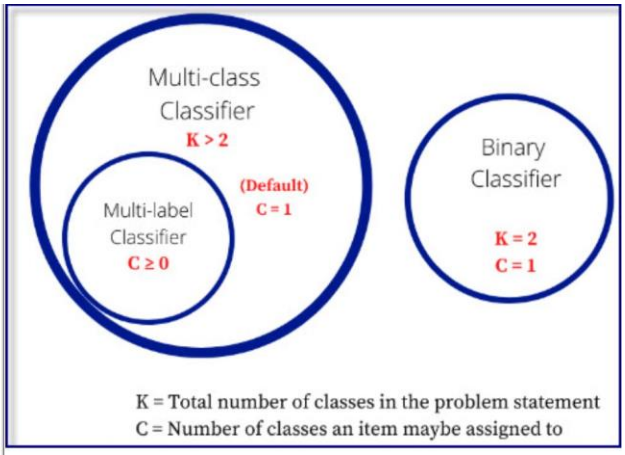
Scikit-Learn provides many helper functions to download popular datasets. MNIST is one of them.

```
>>> from sklearn.datasets import fetch_openml
>>> mnist = fetch_openml('mnist_784', version=1)
>>> mnist.keys()
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'details',
           'categories', 'url'])
```

Google Colab provides free access to a Jupyter Notebook and a small version of the MNIST dataset. Aspiring data scientists can use the platform to practice and experiment with the MNIST dataset. The advantage of using Google Colab is that many popular Python libraries are already pre-installed, making it easy to get started with deep learning and computer vision. The MNIST dataset is also readily available as a sample, allowing users to quickly dive into data science and machine learning.



Scalable multi-output label prediction: From classifier chains to classifier trellises



Multiclass v/s Multilabel classification

Implementation on Google Colab:

https://colab.research.google.com/drive/1nhHVH-gTQnfi3rv7ZS-mjmm8y8Za7uqo?usp=sharing

Execute classification.ipynb:

https://github.com/divyapandey03/Machine-Learning/blob/main/Supervised%20Learning/Classification%20on%20Colab%20using%20MNIST%20dataset/classification.ipynb

Output:

```
[6] %matplotlib inline
    import matplotlib as mpl
    import matplotlib.pyplot as plt

    some_digit = X[0]
    some_digit_image = some_digit.reshape(28, 28)
    plt.imshow(some_digit_image, cmap=mpl.cm.binary)
    plt.axis("off")

    save_fig("some_digit_plot")
    plt.show()

    Saving figure some_digit_plot
```

```
[7] y[0]

    '5'
```

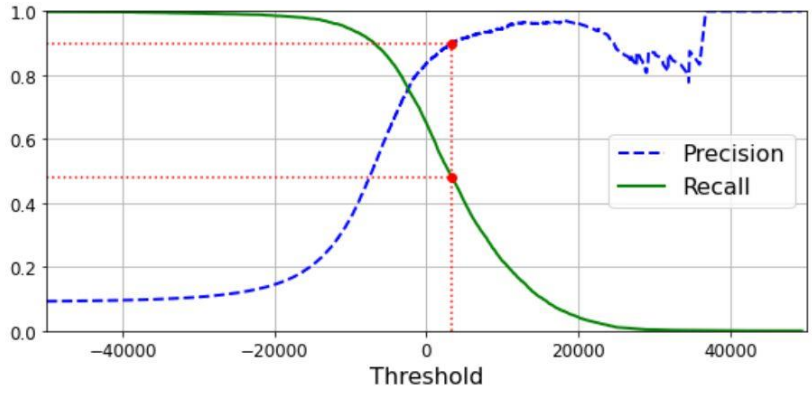
```
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision", linewidth=2)
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall", linewidth=2)
    plt.legend(loc="center right", fontsize=16) # Not shown in the book
    plt.xlabel("Threshold", fontsize=16)        # Not shown
    plt.grid(True)                              # Not shown
    plt.axis([-50000, 50000, 0, 1])             # Not shown



recall_90_precision = recalls[np.argmax(precisions >= 0.90)]
threshold_90_precision = thresholds[np.argmax(precisions >= 0.90)]


plt.figure(figsize=(8, 4))                                                          # Not shown
plot_precision_recall_vs_threshold(precisions, recalls, thresholds)
plt.plot([threshold_90_precision, threshold_90_precision], [0., 0.9], "r:")         # Not shown
plt.plot([-50000, threshold_90_precision], [0.9, 0.9], "r:")                        # Not shown
plt.plot([-50000, threshold_90_precision], [recall_90_precision, recall_90_precision], "r:")# Not shown
plt.plot([threshold_90_precision], [0.9], "ro")                                     # Not shown
plt.plot([threshold_90_precision], [recall_90_precision], "ro")                     # Not shown
save_fig("precision_recall_vs_threshold_plot")                                      # Not shown
plt.show()
```

```
Saving figure precision_recall_vs_threshold_plot
```



```
[38] def plot_precision_vs_recall(precisions, recalls):
         plt.plot(recalls, precisions, "b-", linewidth=2)
         plt.xlabel("Recall", fontsize=16)
         plt.ylabel("Precision", fontsize=16)
         plt.axis([0, 1, 0, 1])
         plt.grid(True)

     plt.figure(figsize=(8, 6))
     plot_precision_vs_recall(precisions, recalls)
     plt.plot([recall_90_precision, recall_90_precision], [0., 0.9], "r:")
     plt.plot([0.0, recall_90_precision], [0.9, 0.9], "r:")
     plt.plot([recall_90_precision], [0.9], "ro")
     save_fig("precision_vs_recall_plot")
     plt.show()
```

```
Saving figure precision_vs_recall_plot
```