CS771 Assignment 2

Yogesh Dudi (201164) Aryan Srivastava (210204)
Anshul Singh Baghel (200155) Priyanshu Nain (200736)
Tarun Beniwal (201046)

1 Solution of Task 1

Our task is to determine whether the HexaDecimal number in the captcha is ODD or EVEN. Since the Python PIL library generated our images, they had no other noise than obfuscating lines.

So our first task was to remove the lines from the images. So we performed following steps:

1. Firstly, we converted images into the HSV color spectrum, which helps us isolate the dominant color information.

2. Now that we have converted it to HSV, we have found the minimum intensity value in the grayscale image, which we will use to threshold the image.

3. Later, we used this minimum value along with a parameter to set a threshold value.

4. Now that we had the threshold value, we converted the pixels, which were lighter than threshold value, to white.

5. Now we converted the image to grayscale and again applied thresholding, which helped us convert all other colors (other than white) to black.

Firstly, we converted images into the HSV color spectrum, which helps us isolate the dominant color information. Firstly, we converted images into the HSV color spectrum, which helps us isolate the dominant color information.

Since we have the image, which contains only digits or alphabets, We first thought to segment all the digits, but later we realized that only the last digit of the hexadecimal number decides whether it is ODD or EVEN. If we have the last digit from ’0’, 2’,’ 4’, ’6’, ‘8, ‘A’, ‘C’, ‘E”, the hexadecimal number will be an EVEN number; otherwise, it will be an ODD number.

So we only needed to segment the last digit of the number.Now that we had only one task left, which was to classify the last digit, So we used various classification algorithms, which are described below:

Preprint. Under review.

We trained a few models and discovered the following observations:

In both classifiers, we divided the data into an 80%–20% train–test.

With an SVM classifier, We used this classifier to classify the PNG image obtained from data processing using **hyperparameters:** Kernel: 'rbf', regularization parameter: 1.0, gamma: 'scale, degree: 3; before processing, we flattened the array of images and used Scikit-Learn's SVM class, such as **SVC** for support vector classification. As a result, we obtained an array of 0s and 1s, 0 for even and 1 for odd decimal values.

With Logistic Regression: Logistic regression is a linear classification model that predicts the probability of an instance belonging to a particular class. It assumes a linear relationship between the input features and the log-odds of the target class.

Iterations: At first we tried to use 100 as the maximum iteration value, but later we found out that for 100 iterations, sometimes it was not converging to optimum. Later, we tried with the value 200, but it was taking too long, So we decided to use 150 as the maximum iteration value, which was best from both perspectives. We used **'lbfgs'**, which stands for Limited-memory Br.den-Fletcher-Goldfarb-Shanno, for optimization. It is an optimization algorithm that approximates the second-order method for efficient training of logistic regression models.

Validation Procedure: We split the dataset into 80%–20% in train to test using the train_test_split function from sklearn.model_selection. After training the model, we calculated validation accuracy using the accuracy_score function from sklearn.metrics.

We obtained an accuracy of 100% on the test and validation sets of data in both the classifiers.

Overall, Logistic regression is computationally more efficient than SVM. It doesn't require complex training procedures or extensive hyperparameter tuning like Advanced Neural Networks, making it relatively straightforward to implement and train.

2 Solution of Task 2

We have submitted predict.py and trained_model.pkl .