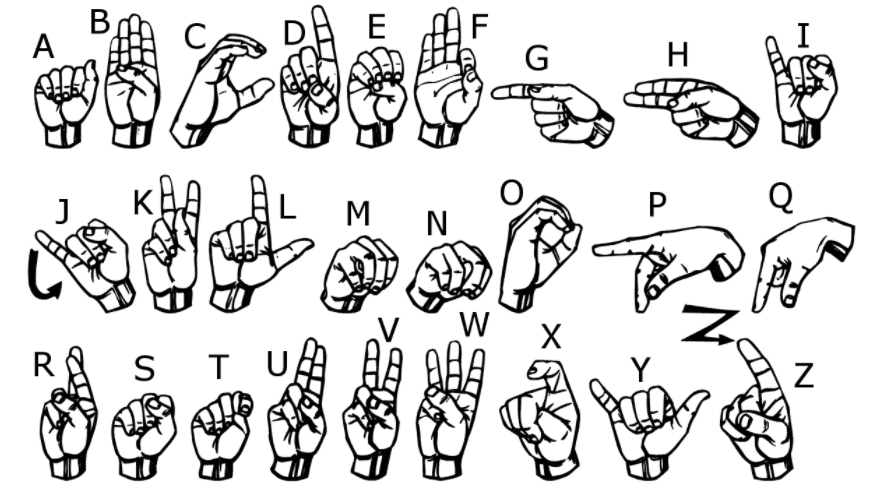
****

**Data Set Information:**

The original [MNIST image dataset](https://en.wikipedia.org/wiki/MNIST_database) of handwritten digits is a popular benchmark for image-based machine learning methods but researchers have renewed efforts to update it and develop drop-in replacements that are more challenging for computer vision and original for real-world applications. As noted in one recent replacement called the Fashion-MNIST [dataset](https://github.com/zalandoresearch/fashion-mnist), the Zalando [researchers](https://arxiv.org/abs/1708.07747) quoted the startling claim that "Most pairs of MNIST digits (784 total pixels per sample) can be distinguished pretty well by just one pixel". To stimulate the community to develop more drop-in replacements, the Sign Language MNIST is presented here and follows the same CSV format with labels and pixel values in single rows. The American Sign Language letter database of hand gestures represent a multi-class problem with 24 classes of letters (excluding J and Z which require motion).

The dataset format is patterned to match closely with the classic MNIST. Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions). The training data (27,455 cases) and test data (7172 cases) are approximately half the size of the standard MNIST but otherwise similar with a header row of label, pixel1,pixel2....pixel784 which represent a single 28x28 pixel image with grayscale values between 0-255. The original hand gesture [image data](https://github.com/mon95/Sign-Language-and-Static-gesture-recognition-using-sklearn) represented multiple users repeating the gesture against different backgrounds. The Sign Language MNIST data came from greatly extending the small number (1704) of the color images included as not cropped around the hand region of interest. To create new data, an image pipeline was used based on ImageMagick and included cropping to hands-only, gray-scaling, resizing, and then creating at least 50+ variations to enlarge the quantity. The modification and expansion strategy was filters ('Mitchell', 'Robidoux', 'Catrom', 'Spline', 'Hermite'), along with 5% random pixelation, +/- 15% brightness/contrast, and finally 3 degrees rotation. Because of the tiny size of the images, these modifications effectively alter the resolution and class separation in interesting, controllable ways.

This dataset was inspired by the Fashion-MNIST [2](https://github.com/zalandoresearch/fashion-mnist) and the machine learning pipeline for gestures by Sreehari [4](https://github.com/mon95/Sign-Language-and-Static-gesture-recognition-using-sklearn).

A robust visual recognition algorithm could provide not only new benchmarks that challenge modern machine learning methods such as Convolutional Neural Nets but also could pragmatically help the deaf and hard-of-hearing better communicate using computer vision applications. The National Institute on Deafness and other Communications Disorders (NIDCD) indicates that the 200-year-old American Sign Language is a complete, complex language (of which letter gestures are only part) but is the primary language for many deaf North Americans. ASL is the leading minority language in the U.S. after the "big four": Spanish, Italian, German, and French. One could implement computer vision in an inexpensive board computer like Raspberry Pi with OpenCV, and some Text-to-Speech to enabling improved and automated translation applications.

**Task -1**

**Please kind the following task regarding the dataset:**

1. Read the dataset
2. Identify **categorical** and **numerical** variables in the dataset and write in *markdown.*
3. **Use all recommended plots will be add-on**
4. Create Dummy Variable for Categorical Data **(n-1 columns)**
5. Split the data into training and testing set (70 % training and 30 % training)
6. Build Logistic regression model with training data.
7. Take default threshold p = 0.5
8. Interpret the results
9. Compute **Confusion Matrix** for training set and testing set
10. From confusion matrix compute
    1. Sensitivity, Specificity, Precession and Accuracy for training set
    2. Sensitivity, Specificity, Precession and Accuracy for testing set
11. Also compute **Kappa Score for** training and testing set(optional)
12. Draw ROC curve Suggest the approx. threshold value for probability of success
13. Plot logistics regression for best threshold probability value which you feel
    1. Compute Confusion matrix
       1. Training data
       2. Testing data
    2. Computer Sensitivity, Specificity, Precession and Accuracy for both training and testing confusion matrix.
14. Any additional recommendation want to made in this model is add-on

Submit the **jupyter notebook** file and also **pdf**  of jupyter files.