Industry Assignment 2 - Machine Learning

Machine learning model to classify any ten letters from one of the Indian languages

Step 1 - Importing Libraries

In [1]:
```
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.optimizers import SGD, Adam
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
```

Step 2 - Importing Datasets

In [2]:
```
data = pd.read_csv("A_Z Handwritten Data.csv").astype('float32')

print(data.head())
```

```
     0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  ...  0.639  0.640  0.641  \
0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...    0.0    0.0    0.0

   0.642  0.643  0.644  0.645  0.646  0.647  0.648
0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
1    0.0    0.0    0.0    0.0    0.0    0.0    0.0
2    0.0    0.0    0.0    0.0    0.0    0.0    0.0
3    0.0    0.0    0.0    0.0    0.0    0.0    0.0
4    0.0    0.0    0.0    0.0    0.0    0.0    0.0

[5 rows x 785 columns]
```

Step 3 - X and Y axis

In [3]:
```
X = data.drop('0', axis = 1)

Y = data['0']
```

Step 4 - Splitting, Training and Testing

In [4]:
```
train_x, test_x, train_y, test_y = train_test_split(X, Y, test_size = 0.2)

train_x = np.reshape(train_x.values, (train_x.shape[0],28, 28))
test_x = np.reshape(test_x.values, (test_x.shape[0],28, 28))
print("Train Data Shape: ", train_x.shape)
print("Test Data Shape: ", test_x.shape)
```

```
Train Data Shape:  (297960, 28, 28)
Test Data Shape:  (74490, 28, 28)
```

Step 5 - Dictionary of Alphabets

In [5]:
```
word_dict = {0:'A', 1:'B', 2:'C', 3:'D', 4:'E', 5:'F', 6:'G', 7:'H', 8:'I', 9:'J', 10:'K', 11:'L', 12:'M',
             13:'N', 14:'O', 15:'P', 16:'Q', 17:'R', 18:'S', 19:'T', 20:'U', 21:'V', 22:'W', 23:'X', 24:'Y', 25:
```

Step 6 - Convert label values into Integer values

In [6]:
```
y_int = np.int0(Y)
count = np.zeros(26, dtype='int')
print(count)

for i in y_int:
    count[i] += 1
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

Step 7 - List named alphabets

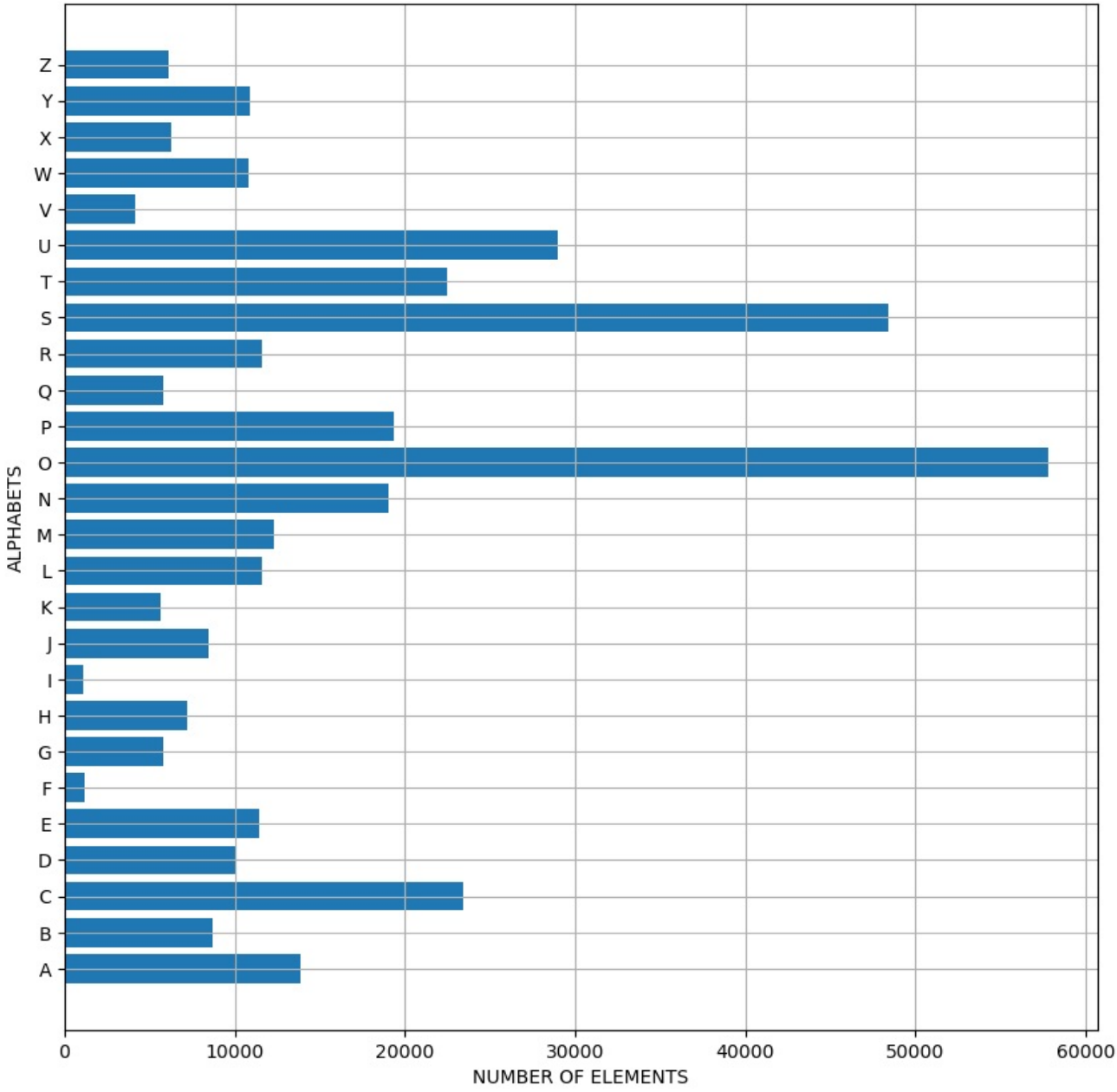
In [7]:
```
alphabets = []
for i in word_dict.values():
    alphabets.append(i)
```

In [8]:
```
alphabets[10]
```

Out[8]:
```
'K'
```

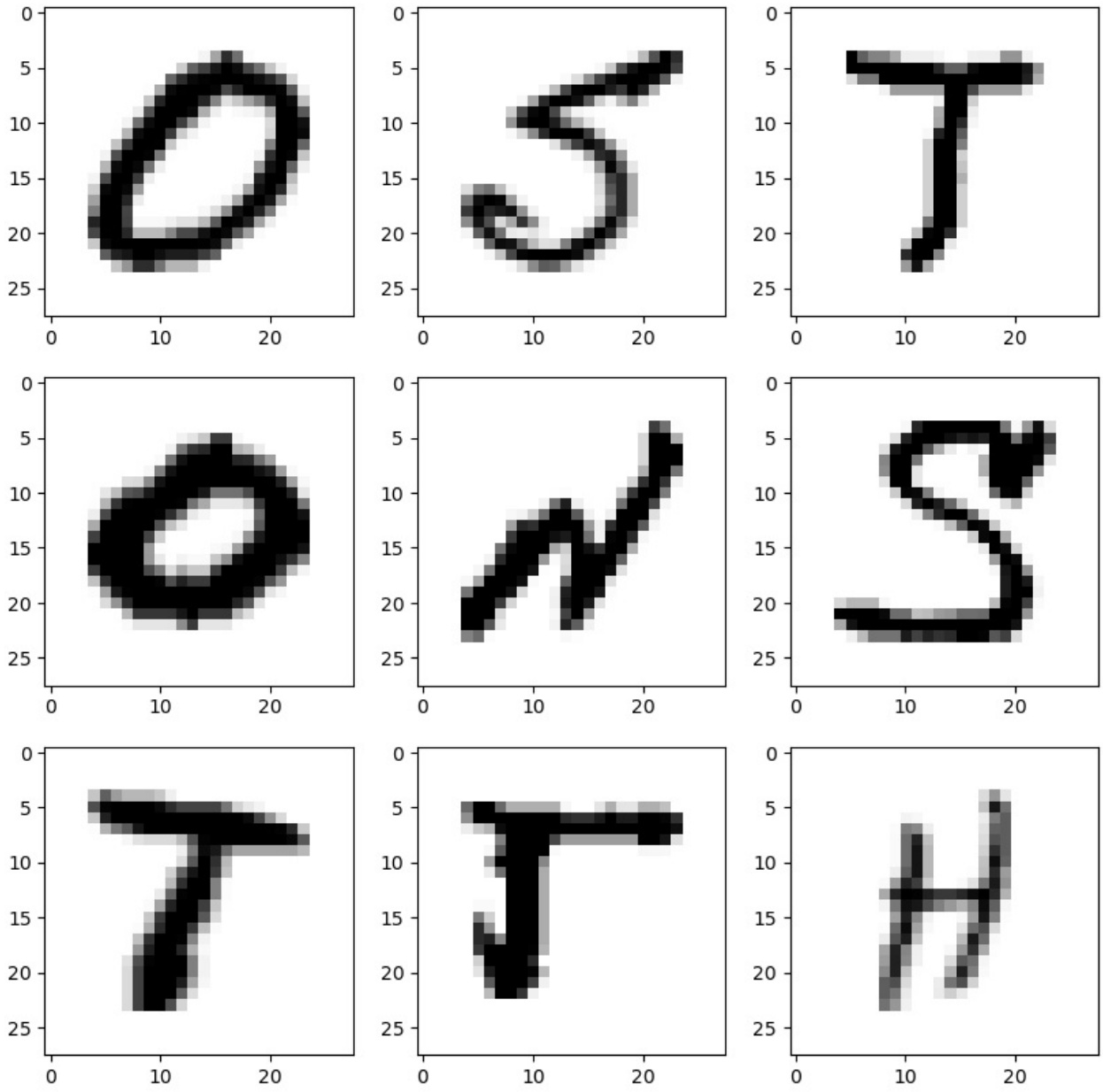
Step 8 - Plotting a Bar Graph

In [9]:
```
fig, ax = plt.subplots(1,1, figsize=(10,10))
ax.barh(alphabets,count)
plt.xlabel("NUMBER OF ELEMENTS")
plt.ylabel("ALPHABETS")
plt.grid()
plt.show()
```



Step 9 - Shuffling the Images

In [10]:
```
shuff = shuffle(train_x[:10])
fig, ax = plt.subplots(3,3, figsize = (10,10))
axes = ax.flatten()
for i in range(9):
    shu = cv2.threshold(shuff[i], 30, 200, cv2.THRESH_BINARY)
    axes[i].imshow(np.reshape(shuff[i], (28,28)), cmap="Greys")
plt.show()
```



Step 10 - Reshaping the Data

In [11]:
```
train_X = train_x.reshape(train_x.shape[0],train_x.shape[1],train_x.shape[2],1)

print("New Shape of the Trained data: ", train_X.shape)
test_X = test_x.reshape(test_x.shape[0], test_x.shape[1], test_x.shape[2],1)
print("New Shape of the Tested data: ", test_X.shape)
```

```
New Shape of the Trained data:  (297960, 28, 28, 1)
New Shape of the Tested data:  (74490, 28, 28, 1)
```

Step 11 - Single Float values to Categorical values

In [12]:
```
train_yOHE = to_categorical(train_y, num_classes=26, dtype='int')
print("New Shape of Train labels : ", train_yOHE.shape)

test_yOHE = to_categorical(test_y, num_classes=26, dtype='int')
print("New Shape of Test labels : ",test_yOHE.shape)
```

```
New Shape of Train labels :  (297960, 26)
New Shape of Test labels :  (74490, 26)
```

Step 12 - Convolutional Layers

In [13]:
```
import tensorflow
import tensorflow as tf
from __future__ import print_function
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras import backend as k
```

```
model = tf.keras.Sequential()
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'valid'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
model.add(Flatten())
model.add(Dense(64,activation ="relu"))
model.add(Dense(128,activation ="relu"))
model.add(Dense(26,activation ="softmax"))
```

Step 13 - Compliling and Fitting the Model

In [14]:
```
model.compile(optimizer = tensorflow.keras.optimizers.Adam(learning_rate=0.001), loss='categorical_crossentropy
              metrics=['accuracy'])
history = model.fit(train_X, train_yOHE, epochs=1, validation_data=(test_X, test_yOHE))
```

```
9312/9312 [==============================] - 158s 17ms/step - loss: 0.1753 - accuracy: 0.9542 - val_loss: 0.091
1 - val_accuracy: 0.9750
```

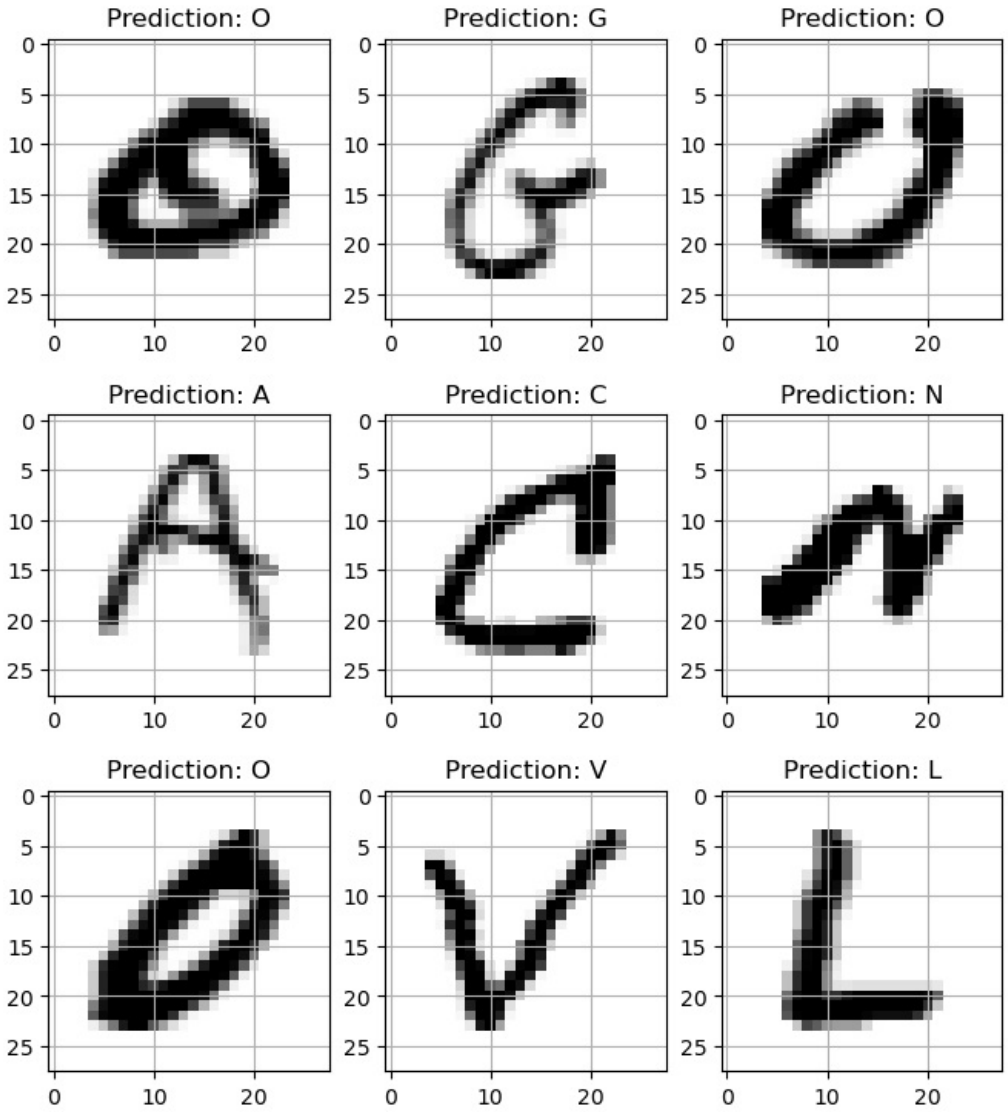
In [15]:
```
model.summary()

model.save(r'model_hand.h5')
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 64)                32832

 dense_1 (Dense)             (None, 128)               8320

 dense_2 (Dense)             (None, 26)                3354

=================================================================
Total params: 137,178
Trainable params: 137,178
Non-trainable params: 0
_________________________________________________________________
```

In [16]:
```
print("The Validation Accuracy is :", history.history['val_accuracy'])

print("The Training Accuracy is :", history.history['accuracy'])

print("The Validation Loss is :", history.history['val_loss'])

print("The Training Loss is :", history.history['loss'])
```

```
The Validation Accuracy is : [0.9750033617019653]
The Training Accuracy is : [0.9541918635368347]
The Validation Loss is : [0.09105141460895538]
The Training Loss is : [0.17526233196258545]
```

In [17]:
```
fig, axes = plt.subplots(3,3, figsize=(8,9))
axes=axes.flatten()

for i,ax in enumerate(axes):
    img = np.reshape(test_X[i], (28,28))
    ax.imshow(img, cmap="Greys")

    pred = word_dict[np.argmax(test_yOHE[i])]
    ax.set_title("Prediction: "+pred)
    ax.grid()
```



In [ ]: