
Por: Jorge Ortiz

# Neural Network: Number Prediction.

This document is presented to recognize digits from 0 - 9, written by hand, it with main aspects to create, train and validate Artificial neural networks in Python with the scikit-learn library.

The first point, it is important to verify that we have all the libraries installed.

## Prerequisites:

Python Libraries.

- Python (versiones >=2.7 o >=3.3)
- Numpy >= 1.8.2 (http://www.numpy.org/)
- SciPy >= 0.13.3 (https://www.scipy.org/)

## What is One Hot Encoding?

A one hot encoding is a representation of categorical variables as binary vectors.

This first requires that the categorical values be mapped to integer values.

Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

## Why Use a One Hot Encoding?

A one hot encoding allows the representation of categorical data to be more expressive.

Many machine learning algorithms cannot work with categorical data directly. The categories must be converted into numbers. This is required for both input and output variables that are categorical.

We could use an integer encoding directly, rescaled where needed. This may work for problems where there is a natural ordinal relationship between the categories, and in turn the integer values, such as labels for temperature ‘cold’, warm’, and ‘hot’.

There may be problems when there is no ordinal relationship and allowing the representation to lean on any such relationship might be damaging to learning to solve the problem. An example might be the labels ‘dog’ and ‘cat’

In these cases, we would like to give the network more expressive power to learn a probability-like number for each possible label value. This can help in both making the problem easier for the network to model. When a one hot encoding is used for the output variable, it may offer a more nuanced set of predictions than a single label.

# Setup:

The installation of **scikit-learn** can be done easily through the following command:

```
pip install -U scikit-learn
```

Where the option **- U** indicates that if the package exists, it must be updated to the last existing stable version.

Similarly, if more details are desired, it is feasible to consult the following enlace (http://scikit-learn.org/stable/install.html).

# Deep Neural Networks

Deep-learning networks are distinguished from the more commonplace single-hidden-layer neural networks by their depth; that is, the number of node layers through which data must pass in a multistep process of pattern recognition.

Earlier versions of neural networks such as the first perceptrons were shallow, composed of one input and one output layer, and at most one hidden layer in between. More than three layers (including input and output) qualifies as “deep” learning. So deep is not just a buzzword to make algorithms seem like they read Sartre and listen to bands you haven’t heard of yet. It is a strictly defined term that means more than one hidden layer.

In deep-learning networks, each layer of nodes trains on a distinct set of features based on the previous layer’s output. The further you advance into the neural net, the more complex the features your nodes can recognize, since they aggregate and recombine features from the previous layer.

1) We define utilities to classify the patterns of ones and zeros. In addition to the reading of the corpus where the writing patterns of several individuals that make up the 0 - 9 are found.

In [1]:

```
import re #Expresiones regulares
import itertools

class Utilities:

    def __init__(self, path = 'corpus/digits-database.data'):
        self.path = path
        self.regex = re.compile('(0|1){2,}') # Patrones pares de 0 y unos
        self.regexno = re.compile('(\s)+[0-9]{1}') # Busca un unico numero el cu
al tenga un espacio o tabulacion antes del mismo.


    def generate_indices(self):
        _dict = []
        with open(self.path, 'r') as _f: #abre el archivo corpus
            pivote = 0
            flag = False
            lineno = 0
            for line in _f:
                if self.regex.match(line)!=None and not flag:
                    pivote = lineno
                    flag = True
                if self.regexno.match(line)!=None and flag:
                    _dict.append((int(line.replace(' ','')),pivote,lineno))
                    flag = False
                lineno += 1
            _f.close()

        return _dict

    def get_digit(self,_slice, _end):
        data = []
        with open(self.path, 'r') as _f:
            for line in itertools.islice(_f, _slice, _end):
                data.append([int(i) for i in line.lstrip().rstrip()])

            _f.close()
        return data
```

2) The training is done. Number of interactions: 500.

In [2]:

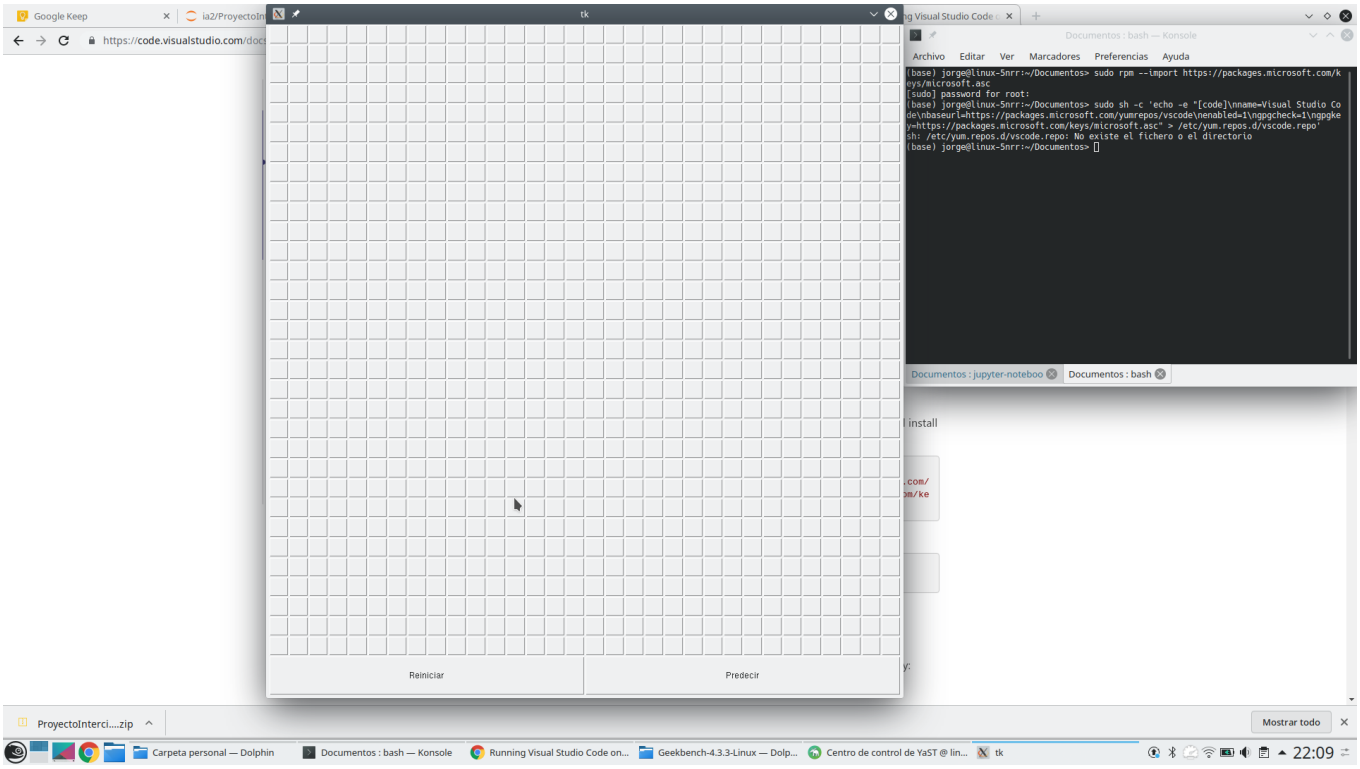
```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from tkinter import *
from tkinter import ttk
import tkinter.messagebox as msg
import numpy as np


utilities = Utilities()

class Ventana(Frame):

    def __init__(self, master = None):
        super().__init__(root)
        self.master = master
        self.coordenadas = [] #Almacena la matriz que se recupera de la interfaz
        self.utilities = Utilities()
        self.indices = self.utilities.generate_indices()
        self.n = []
        self.entrada = []
        self.datos = []
        self.delta = []
        self.init() #llama al init para entrenar la red.

    def normalizador(self):
        for j, k, l in self.indices:
            self.n.append(j)
            self.entrada.append((k,l))

        for i in range(0, len(self.indices)):
            inicio, fin = self.entrada[i]
            fila = np.ravel(np.matrix(self.utilities.get_digit(inicio, fin)))
            self.datos.append(fila)
            self.delta.append(self.n[i])

    def init(self):
        self.master.resizable(0, 0)
        self.grid(row = 0,column = 0)
        self.matriz()
        self.normalizador()
        self.train() #Llama al metodo para entrenar la red

        btnReiniciar = Button(self, text="Reiniciar", height=3, command=self.rei
niciar) #Limpia la grilla
        btnReiniciar.grid(columnspan = 16, sticky = W + E + N + S,row = 32, colu
mn = 0)

        btnPredecir = Button(self, text="Predecir", height=3, command=self.decod
e)
        btnPredecir.grid(columnspan = 16, sticky = W + E + N + S,row = 32, colum
n = 16)

    def train(self): #entrena la red
        self.label_encoder = LabelEncoder()
        salida = self.label_encoder.fit_transform(self.delta)
        onehot_encoder = OneHotEncoder(sparse=False)
        salida = salida.reshape(len(salida), 1)
```

```
        self.onehot_encoded = onehot_encoder.fit_transform(salida)
        x_train, x_test, d_train, d_test = train_test_split(self.datos, self.one
hot_encoded, test_size=0.80, random_state=0)
        self.mlp = MLPClassifier(solver = 'lbfgs', activation='logistic', verbos
e=True, alpha=1e-4, tol=1e-15, max_iter=500, \
        hidden_layer_sizes=(1024, 800, 400, 200, 10))
        self.mlp.fit(self.datos, self.onehot_encoded)

        prediccion = (np.argmax(self.mlp.predict(x_test), axis = 1) + 1).reshape
(-1, 1)
        matriz = confusion_matrix((np.argmax(d_test, axis = 1) + 1).reshape(-1,
1), prediccion)
        print(matriz)

    def decode(self):
        entrada = self.normaliza(32, self.coordenadas)
        numero = np.ravel(np.matrix(entrada))
        res = self.mlp.predict(numero.reshape(1, -1)) #Red ya entrenada
        num = (np.argmax(res, axis=1)+1).reshape(-1, 1)
        aux = []
        matriz = []
        resultado = int(num[0] - 1)
        print(resultado)
        return resultado

    def matriz(self):
        self.btn = [[0 for x in range(32)] for x in range(32)]
        for x in range(32):
            for y in range(32):
                self.btn[x][y] = Button(self, command=lambda x1=x, y1=y: self.di
bujar(x1,y1))
                self.btn[x][y].grid(column = x, row = y)

    def normaliza(self, n, coordenadas): #Transforma la interfaz de botones en u
na matriz
        matriz = []
        for i in range(n):
            matriz.append([0 for j in range(n)])

        for i in range(len(coordenadas)):
            x, y = coordenadas[i]
            matriz[y][x] = 1
        return matriz

    def dibujar(self, x, y):
        self.btn[x][y].config(bg = "black")
        self.coordenadas.append((x, y))

    def reiniciar(self):
        self.matriz()
        self.coordenadas = [] #vacia la matriz

if __name__ == '__main__':
    root = Tk()
    ventana = Ventana(root)
    root.mainloop()
```

```
/home/jorge/anaconda3/lib/python3.7/site-packages/sklearn/preprocess
ing/_encoders.py:371: FutureWarning: The handling of integer data wi
ll change in version 0.22. Currently, the categories are determined
based on the range [0, max(values)], while in the future they will b
e determined based on the unique values.
If you want the future behaviour and silence this warning, you can s
pecify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert
the categories to integers, then you can now use the OneHotEncoder d
irectly.
  warnings.warn(msg, FutureWarning)
```

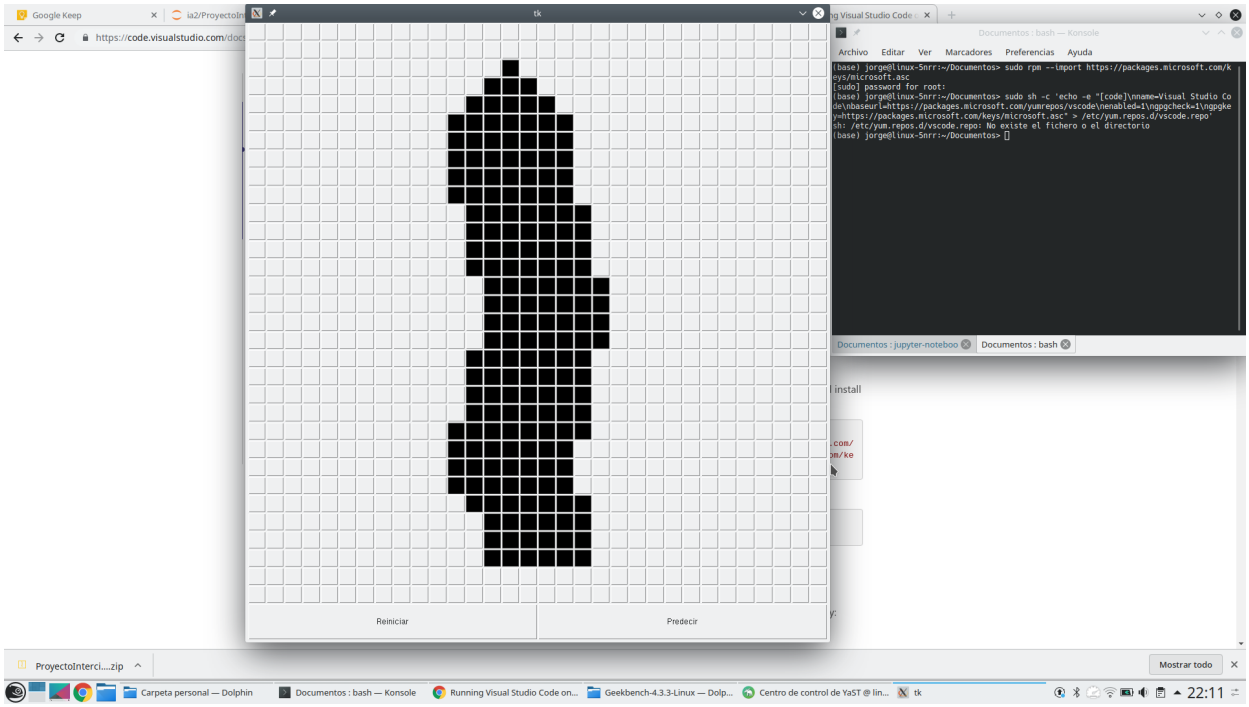
```
[[64  0  0  0  0  0  0  0  0  0]
 [ 1 77  0  0  0  0  0  0  0  1]
 [ 0  0 72  0  0  0  0  0  0  1]
 [ 0  0  0 73  0  0  0  0  0  0]
 [ 0  1  0  0 84  2  0  0  0  0]
 [ 1  0  0  0  1 87  0  0  0  0]
 [ 0  0  0  0  1  0 66  0  0  0]
 [ 0  0  0  0 78  0  0  0  0  0]
 [ 0  0  0  0  1  0  1  0 72  0]
 [ 1  0  0  0  0  1  0  0  0 71]]
```

A blank window will open, this window contains buttons so we will press it.

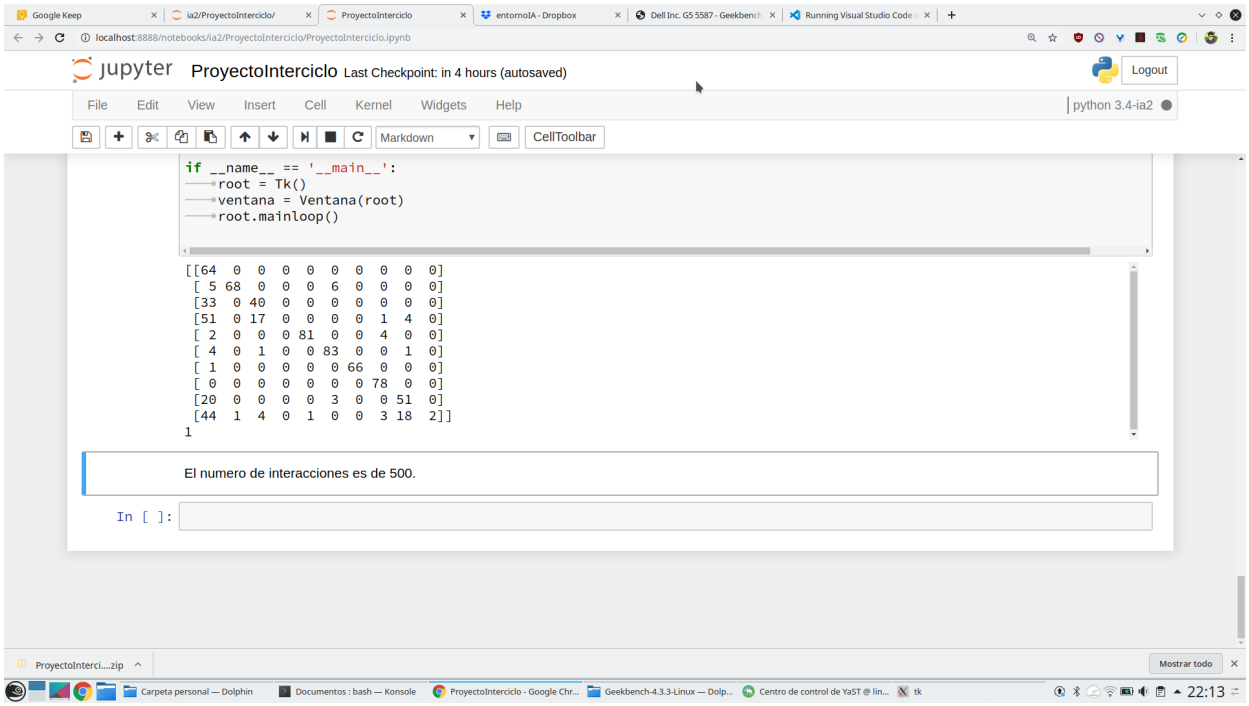


# Adding new tests

- The first test is to write the number 1

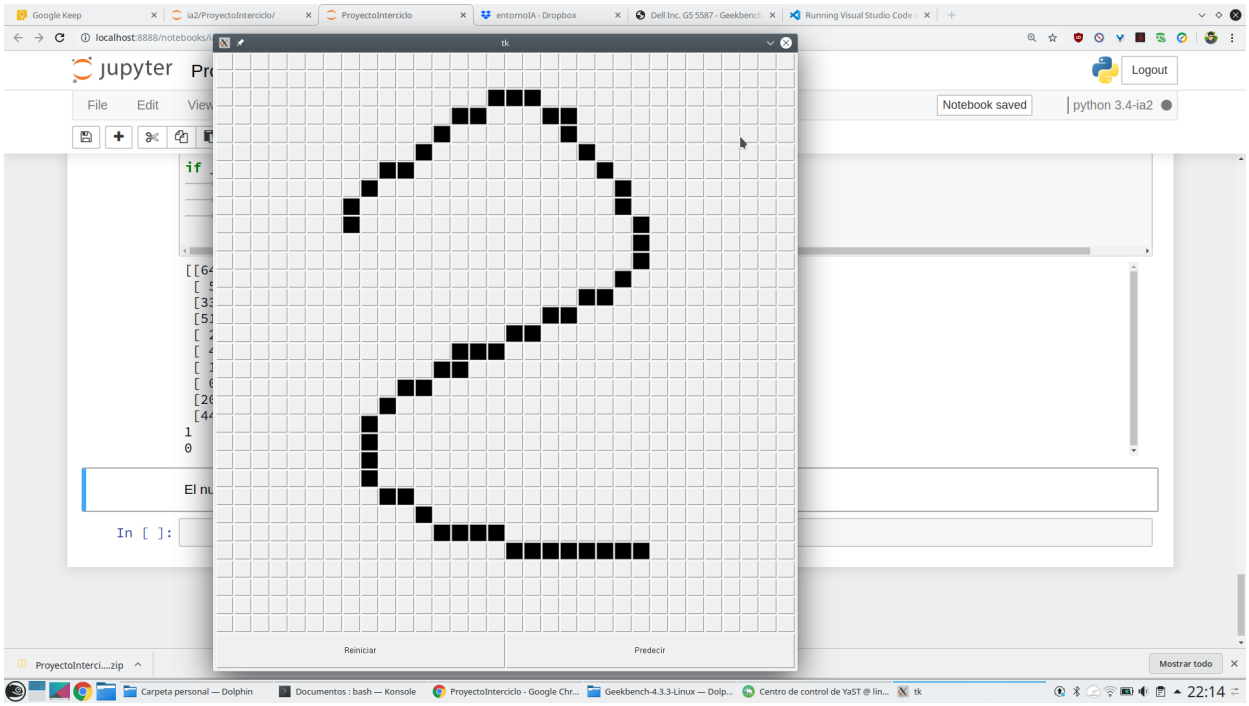



It checks in the console that the desired number offers.

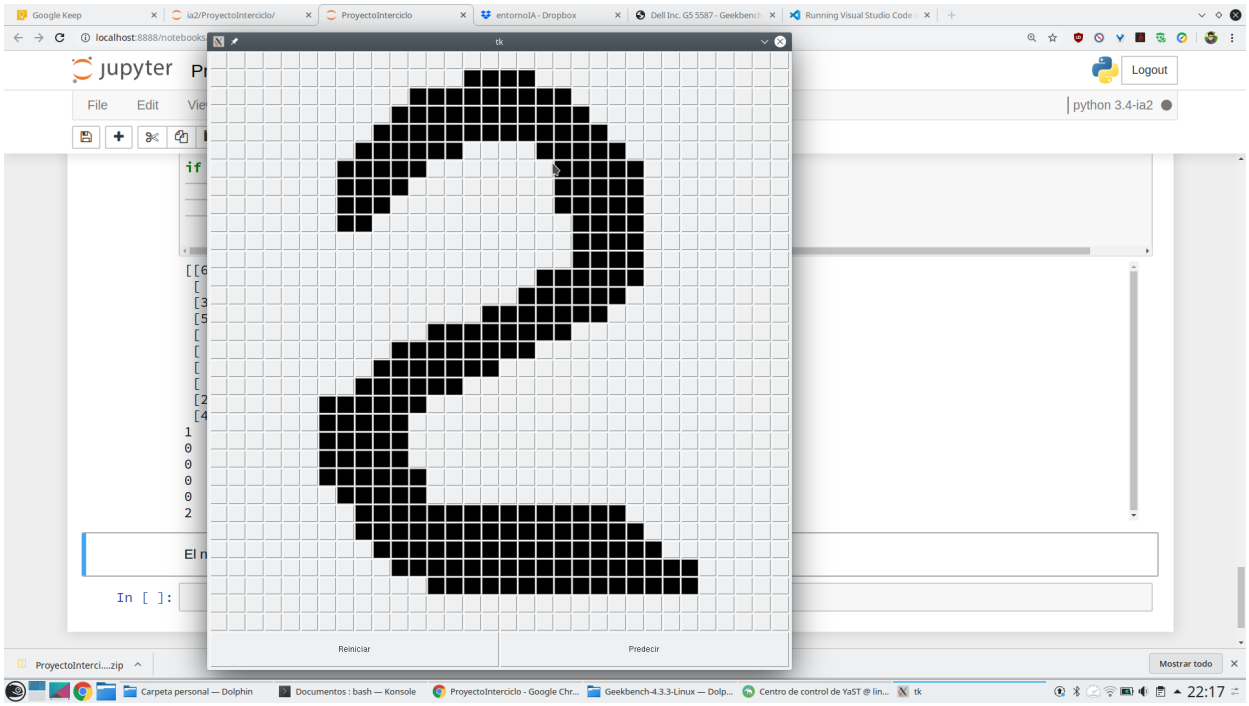



Correct prediction

- The second test is to write the number 2

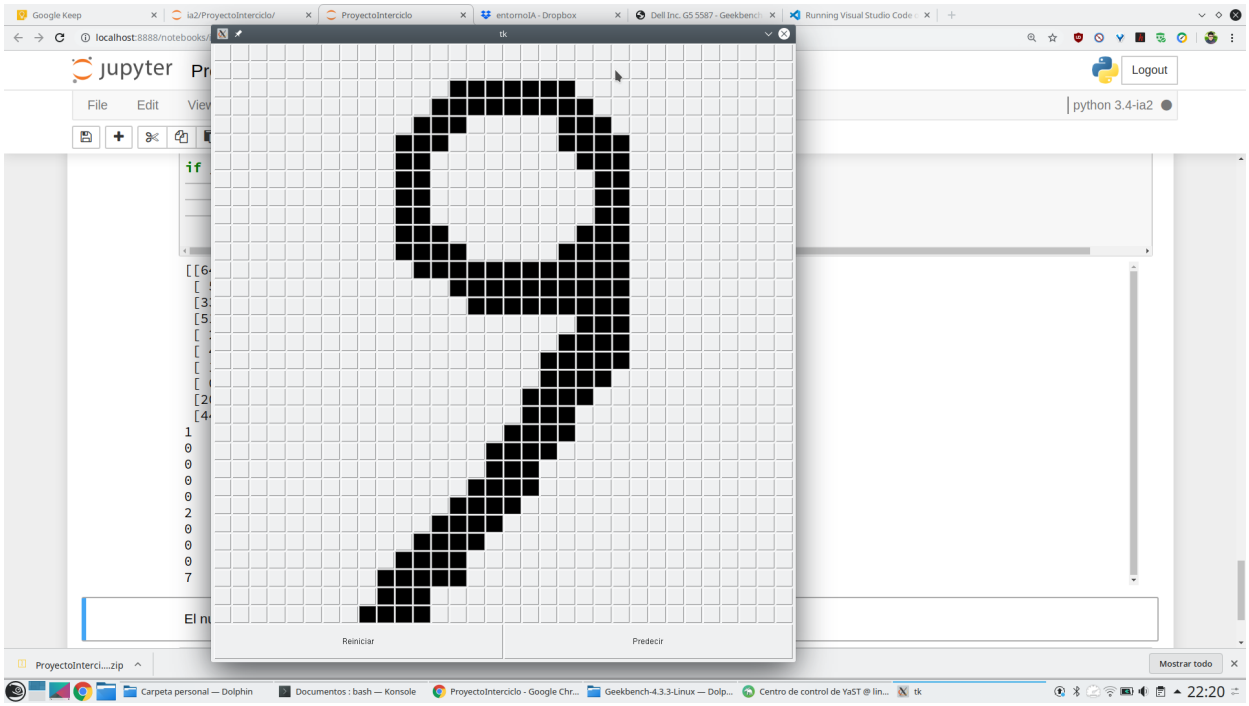



We verify in the console that the desired number offers 0, for which it is wrong, we will add more points to obtain better precision.

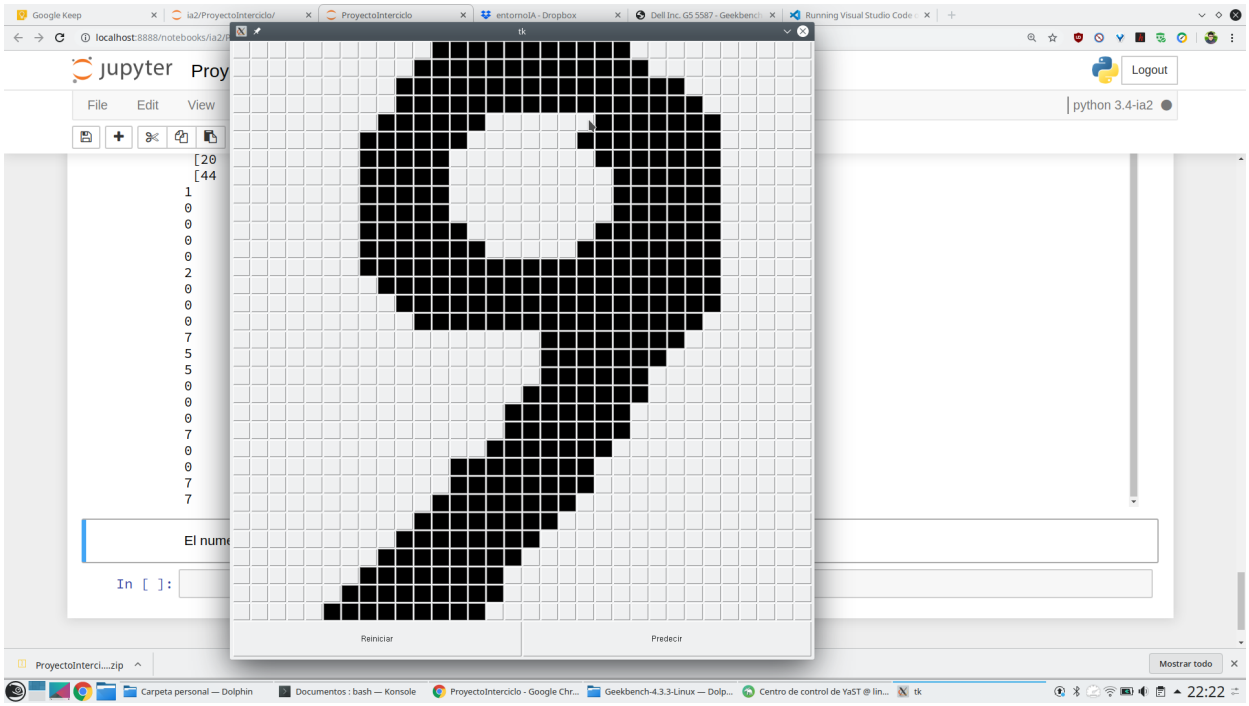



Correct prediction.

- The third test is to write the number 9




The prediction is incorrect, print 7 instead of 9




The prediction failed.

# CONCLUSION.

- The more training interactions you could get better results, but it is not always the case, it reaches such a point that there is no improvement.

# References


- scikit-learn(2019). Recovered from: https://scikit-learn.org/stable/ (https://scikit-learn.org/stable/)
- Browlin, J. 2017. How to One Hot Encode Sequence Data in Python. Recovered from: https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/ (https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/)


In [ ]: