Trabajo integrador - Parte 1

Python y Numpy

Nombre:

```
import numpy as np
```

Ejercicio 1

Dada una matriz en formato *numpy array*, donde cada fila de la matriz representa un vector matemático, se requiere computar las normas l_0, l_1, l_2, l_∞, según la siguientes definiciones:

\begin{equation} ||\mathbf{x}||^{p} = \bigg(\sum_{j=1}^{n}{|x_i|^p}\bigg)^{\frac{1}{p}} \ end{equation}

con los casos especiales para $p=0$ y $p=\infty$ siendo:

\begin{equation} \begin{array}{rcl} ||\mathbf{x}||*0 & = & |bigg(|sum*{j=1 \wedge x_j != 0}{|x_i|}\ bigg)\ ||\mathbf{x}||*{|infty} & = & |max*{i}{|x_i|}\ \end{array} \end{equation}

```
def norm_lp(X,p):
    if(p==0):
        return np.sum(X!=0, axis=1)
    if(p<0):
        return np.max(X, axis=1)

    return (np.sum(np.abs(X**p), axis=1)** 1/p)

x = np.array([[1,0,3,0],[0,3,5,6],[1,2,3,5]], dtype=int)

print("norm_l0 = " +str(norm_lp(x,0)))
print("\nnorm_l1 = " +str(norm_lp(x,1)))
print("\nnorm_l2 = " +str(norm_lp(x,2)))
print("\nnorm_l-inf = " +str(norm_lp(x,-1)))  # -1 para indicar norma
infinita
```

```
norm_l0 = [2 3 4]

norm_l1 = [ 4. 14. 11.]

norm_l2 = [ 5.  35.  19.5]

norm_l-inf = [3 6 5]
```

Ejercicio 2

En clasificación contamos con dos arreglos, la “verdad” y la “predicción”. Cada elemento de los arreglos pueden tomar dos valores, “True” (representado por 1) y “False” (representado por 0). Entonces podemos definir 4 variables:

- True Positive (TP): El valor verdadero es 1 y el valor predicho es 1
- True Negative (TN): El valor verdadero es 0 y el valor predicho es 0
- False Positive (FP): El valor verdadero es 0 y el valor predicho es 1
- False Negative (FN): El valor verdadero es 1 y el valor predicho es 0

A partir de esto definimos:

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- Accuracy = (TP + TN) / (TP + TN + FP + FN)

Calcular las 3 métricas con Numpy y operaciones vectorizadas.

```
import numpy as np

def get_tp(truth,pred):
    return np.sum((truth==1)*(pred==1))

def get_tn(truth,pred):
    return np.sum((truth==0)*(pred==0))

def get_fn(truth,pred):
    return np.sum((truth==1)*(pred==0))

def get_fp(truth,pred):
    return np.sum((truth==0)*(pred==1))

def precision(truth,pred):
    tp = get_tp(truth,pred)
    fp = get_fp(truth,pred)
    return tp / (tp + fp)

def recall(truth,pred):
    tp = get_tp(truth,pred)
    fn = get_fn(truth,pred)
    return tp / (tp + fn)

def accuracy2(truth,pred):
    tp = get_tp(truth,pred)
    tn = get_tn(truth,pred)
    fp = get_fp(truth,pred)
    fn = get_fn(truth,pred)
    return (tp + tn) / (tp + tn + fp + fn)
```

```
def accuracy(truth,pred):
    return np.sum(truth==pred) / truth.shape[0]

truth = np.array(     [1,1,0,1,1,1,0,0,0,1])
prediction = np.array([1,1,1,1,0,0,1,1,0,0])

print('precision = ' + str(precision(truth,prediction)))
print('recall = ' + str(recall(truth,prediction)))
print('accuracy = ' + str(accuracy(truth,prediction)))
print('accuracy2 = ' + str(accuracy2(truth,prediction)))
```

```
precision = 0.5
recall = 0.5
accuracy = 0.4
accuracy2 = 0.4
```

Ejercicio 3

Crear una función que separe los datos en train-validation-test. Debe recibir de parametros:

- X: Array o Dataframe que contiene los datos de entrada del sistema.
- y: Array o Dataframe que contiene la(s) variable(s) target del problema.
- train_percentage: *float* el porcentaje de training.
- test_percentage: *float* el porcentaje de testing.
- val_percentage: *float* el porcentaje de validación.
- shuffle: *bool* determina si el split debe hacerse de manera random o no.

Hints:

- Usar Indexing y slicing
- Usar np.random.[...]

```
def split(X_input,
          Y_input,
          val_size=0.15,
          test_size=0.15,
          random_state=42,
          shuffle=True):

      _X_input = np.copy(X_input)
      _Y_input = np.copy(Y_input)

      if not _X_input.shape[0] == _Y_input.shape[0]:
            raise ValueError("Los datos (X_input, Y_input) tienen
distintas longitudes.")

      train_size = 1 - test_size - val_size

      if(train_size<0):
            raise ValueError("El porcentaje de datos de validacion y
```

```
test no puede ser mayor al 100%.")

      # Mezclar los datos de manera tal que ambos conserven los mismos
indices
      if(shuffle):
            np.random.seed(random_state)
            ran_idx = np.random.permutation(len(_X_input))
            _X_input = _X_input[ran_idx]
            _Y_input = _Y_input[ran_idx]

      total_len = _X_input.shape[0]
      train_len = int(train_size*total_len)
      val_len = int(val_size*total_len)

      X_train = np.array(_X_input[0:train_len])
      X_val = np.array(_X_input[train_len:train_len+val_len])
      X_test = np.array(_X_input[train_len+val_len:total_len])

      Y_train = np.array(_Y_input[0:train_len])
      Y_val = np.array(_Y_input[train_len:train_len+val_len])
      Y_test = np.array(_Y_input[train_len+val_len:total_len])

      return X_train, X_val, X_test, Y_train, Y_val, Y_test


X = np.array([[0,1,2,3,4,5,6,7,8,9],[10,11,12,13,14,15,16,17,18,19],
[20,21,22,23,24,25,26,27,28,29]]).T
Y = np.array([100,101,102,103,104,105,106,107,108,109])

X_train, X_val, X_test, Y_train, Y_val, Y_test = split(X, Y,
shuffle=False)
print("X_train: \n"+str(X_train)+"\n")
print("X_val: \n"+str(X_val)+"\n")
print("X_test: \n"+str(X_test)+"\n")
print("Y_train: \n"+str(Y_train)+"\n")
print("Y_val: \n"+str(Y_val)+"\n")
print("Y_test: \n"+str(Y_test)+"\n")

X_train:
[[ 0 10 20]
 [ 1 11 21]
 [ 2 12 22]
 [ 3 13 23]
 [ 4 14 24]
 [ 5 15 25]
 [ 6 16 26]]

X_val:
[[ 7 17 27]]
```

```
X_test:
[[ 8 18 28]
 [ 9 19 29]]

Y_train:
[100 101 102 103 104 105 106]

Y_val:
[107]

Y_test:
[108 109]
```

```
X_train, X_val, X_test, Y_train, Y_val, Y_test = split(X, Y,
shuffle=True)
print("X_train: \n"+str(X_train)+"\n")
print("X_val: \n"+str(X_val)+"\n")
print("X_test: \n"+str(X_test)+"\n")
print("Y_train: \n"+str(Y_train)+"\n")
print("Y_val: \n"+str(Y_val)+"\n")
print("Y_test: \n"+str(Y_test)+"\n")
```

```
X_train:
[[ 8 18 28]
 [ 7 17 27]
 [ 1 11 21]
 [ 0 10 20]
 [ 6 16 26]
 [ 4 14 24]
 [ 5 15 25]]

X_val:
[[ 3 13 23]]

X_test:
[[ 9 19 29]
 [ 2 12 22]]

Y_train:
[108 107 101 100 106 104 105]

Y_val:
[103]

Y_test:
[109 102]
```