

# ALGORITMOS DE APRENDIZAJE DE MÁQUINA APLICADOS A LA GENERACIÓN DE MODELOS PREDICTIVOS

---

Ing. José Navas Sú  
Escuela de Ingeniería en Computación

# Agenda

1. Investigación
2. Calendarización de Modelos en Paralelo

# 1. INVESTIGACIÓN

---

# 1.1 Definición del Problema

## PROBLEMA

Ausencia de modelos prácticos y herramientas automatizadas que ayuden a identificar de manera oportuna estudiantes que podrían requerir intervención preventiva por parte del Tec.

TecDigital

### TÍTULO DE LA TESIS

Modelo Predictivo de Exito Académico  
Aplicando Algoritmos de Aprendizaje de Máquina  
sobre Interacciones en el TecDigital

### ALCANCE DE LA TESIS

- Conjunto de datos: interacciones e historial académico
- Algoritmos considerados: NN, SVM y Linear Regression
- El modelo predictivo permitirá proyectar éxito académico

## EXTRACCIÓN Y PREPROCESO DE LOS DATOS

- Extracción de interacciones e historial académico de estudiantes 2016
- Filtrado de interacciones y cursos:
  - No estudiantes
  - Cursos no finalizados: retiros, congelamiento, incompletos
  - Cursos que no aplican:  
no activos, no semestrales, no bachillerato o licenciatura continua
  - Eliminación de casos extremos
- Consolidación y categorización de interacciones  
(*File Storage 49 %, dotLRN 39 %, Evaluation 8 %*)

# 1.4 Modelo de Datos

Cuadro: Categorías empleadas por el módulo TAM

Id	Categoría	Interacciones	%
7	Documentos	5,533,101	48.80
11	dotLRN (core)	4,383,740	38.70
5	Evaluaciones	904,938	7.99
8	Noticias	192,486	1.70
12	Foros	164,735	1.45
14	Tareas	74,071	0.65
16	Calendario	28,581	0.25
10	GAAP (Activ. Aprend.)	20,174	0.18
1	Cuestionarios	14,511	0.13
15	Adjuntos	4,695	0.04
4	Asistencia	2,259	0.02
6	Preguntas Frecuentes	1,533	0.01
3	Encuestas	1,442	0.01
17	Mensajería SMS	292	0.00
2	Chat	8	0.00
9	Redes Sociales	8	0.00
13	Gener. Diseño Instrucc.	6	0.00



## 1.4 Modelo de Datos

### CONJUNTO DE DATOS

- 204 variables (12 semanas  $\times$  17 categorías): [0..5000]
- Variable respuesta: [0-Aprobó, 1-Reprobó] (68 % ~ 32 %)
- Cantidad de muestras: 84,808 (11,326,580 interacciones)
- Modelos a ajustar: 10 (semanas [1,2,3] hasta [1,2,3,...,12])
- Normalización:  $\frac{X - \min X}{\max X - \min X}$

# 1.5 Diseño y Comparación de Algoritmos

## PLATAFORMA COMPUTACIONAL

- Cluster Kabré del CeNAT, infraestructura HPC
- Cuenta con 2 tipos de colas y nodos:
  - **5 Cadejos:** Intel Xeon E5530/8cores/16thread/32GBram/2 Tesla-C1060
  - **20 Zarate:** Intel Xeon Phi/64cores/256thread/96GBram
  - **4 Tule:** Intel Xeon E3-1225/4cores/4thread/16GBram/1 Tesla-K40
- Sistema de colas Torque, calendarizador Maui
- Módulos instalados en el cluster utilizados:
  - R y biblioteca *glmnet* para Regresión Logística
  - Python y *libsvm* escrita en C y Python para Máquinas de Soporte Vectorial
  - Python, Keras y TensorFlow para Redes Neuronales

## 1.6 Ajuste y Validación de Modelos Predictivos

Cuadro: Ajuste de modelos Seleccionados con Redes Neuronales

Se *	Do *	Ni *	Ne *	Ep *	Bs *	Exactitud
3	0.150	12	550	800	256	0.683438853715
4	0.050	5	350	2500	128	0.682772409197
5	0.075	15	550	2500	128	0.681772742419
6	0.075	12	350	2000	512	0.695434855048
7	0.075	12	350	2200	512	0.747417527491
8	0.075	5	450	2500	256	0.771439520163
9	0.075	5	350	2200	256	0.758413862046
10	0.050	5	250	1600	128	0.754748417194
11	0.150	9	450	2500	512	0.762745751416
12	0.150	12	550	1600	512	0.783405531494

\* Se=Semana, Do=Dropout, Ni=Niveles Ocultos, Ne=Neuronas, Ep=Epoas, Bs=Tamaño de Batch

# 1.6 Ajuste y Validación de Modelos Predictivos

Cuadro: Calidad de Exactitud en Modelos Seleccionados

Semana	$F_1$ -Score	Precisión	Recuperación
3	0.022633744856	0.733333333333	0.011494252874
4	0.015117830146	1.000000000000	0.007616487455
5	0.021248339973	0.888888888889	0.010752688172
6	0.241322314052	0.577075098814	0.152560083595
7	0.579643473267	0.644501278772	0.526645768025
8	0.562060889935	0.639147802929	0.501567398119
9	0.744905855043	0.877811550152	0.646953405018
10	0.575575575576	0.552353506244	0.600835945664
11	0.632786885246	0.663230240557	0.605015673981
12	0.645948945616	0.688757396452	0.608150470219

## 1.6 Ajuste y Validación de Modelos Predictivos

### MÉTRICA $F_1$ -SCORE

$$P = \frac{V^+}{V^+ + F^+}, \text{ (precisión)}$$

Fracción de estudiantes identificados correctamente como reprobados de todos los identificados como reprobados por el modelo.

$$R = \frac{V^+}{V^+ + F^-}, \text{ (recuperación)}$$

Fracción de estudiantes identificados correctamente como reprobados del total real de reprobados.

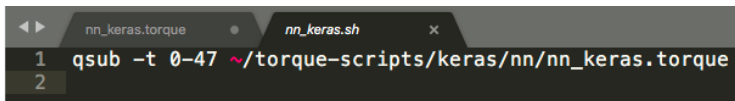
$$F_1 = \frac{2PR}{P+R}$$

Balance entre precisión y recuperación. Si la precisión, o la recuperación, o ambas, son muy bajas,  $F_1$  será bajo. Se escoge los modelos con valores para  $F_1$  mayores.

## 2. CALENDARIZACIÓN DE MODELOS EN PARALELO

---

## 2. Calendarización de Modelos en Paralelo



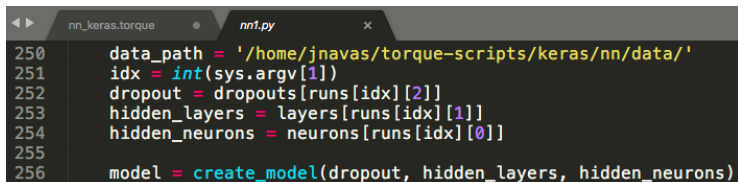
```
nn_keras.torque  nn_keras.sh x
1 qsub -t 0-47 ~/torque-scripts/keras/nn/nn_keras.torque
2
```

## 2. Calendarización de Modelos en Paralelo

```
nn_keras.torque
1 #PBS -N nn_keras
2 ## merge standard error and output
3 #PBS -j oe
4 ## direct streams to our logfile
5 #PBS -o nn_keras-$PBS_ARRAYID-$PBS_JOBID.log
6 #PBS -q phi-n6h96
7 #PBS -l nodes=1:ppn=32
8 #PBS -l walltime=96:00:00
9
10 DIR=~/.torque-scripts/keras/nn
11
12 #alias
13 source ~/.bashrc
14
15 cd $DIR
16
17 module load intelpython/2.7
18
19 python nn1.py $PBS_ARRAYID
20
```



## 2. Calendarización de Modelos en Paralelo



The image shows a code editor window with two tabs: 'nn\_keras.torque' and 'nn1.py'. The 'nn1.py' tab is active, displaying Python code. The code defines variables for data path, index, dropout, hidden layers, and hidden neurons, and then creates a model using a function called 'create\_model'.

```
250 data_path = '/home/jnavas/torque-scripts/keras/nn/data/'
251 idx = int(sys.argv[1])
252 dropout = dropouts[runs[idx][2]]
253 hidden_layers = layers[runs[idx][1]]
254 hidden_neurons = neurons[runs[idx][0]]
255
256 model = create_model(dropout, hidden_layers, hidden_neurons)
```

## 2. Calendarización de Modelos en Paralelo

```
nn_keras.torque  nn1.py x
36 dropouts = [0.05,0.1,0.15]
37 layers   = [5,10,15,20]
38 neurons  = [250,300,350,500]
39
40 runs = [
41     [0,0,0],[0,0,1],[0,0,2],
42     [0,1,0],[0,1,1],[0,1,2],
43     [0,2,0],[0,2,1],[0,2,2],
44     [0,3,0],[0,3,1],[0,3,2],
45     [1,0,0],[1,0,1],[1,0,2],
46     [1,1,0],[1,1,1],[1,1,2],
47     [1,2,0],[1,2,1],[1,2,2],
48     [1,3,0],[1,3,1],[1,3,2],
49     [2,0,0],[2,0,1],[2,0,2],
50     [2,1,0],[2,1,1],[2,1,2],
51     [2,2,0],[2,2,1],[2,2,2],
52     [2,3,0],[2,3,1],[2,3,2],
53     [3,0,0],[3,0,1],[3,0,2],
54     [3,1,0],[3,1,1],[3,1,2],
55     [3,2,0],[3,2,1],[3,2,2],
56     [3,3,0],[3,3,1],[3,3,2]
57 ]
58
```

¡Muchas Gracias!