Machine Learning con Python

Machine Learning with Python

Autor: Juan Camilo Varón
Departamento de sistemas, Universidad Tecnológica de Pereira, Colombia
Correo-e: juan.varon@utp.edu.co

***Resumen*— Machine Learning es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente. Aprender en este contexto quiere decir identificar patrones complejos en millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros. Automáticamente, también en este contexto, implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana.**

En muchas ocasiones el campo de actuación del aprendizaje automático se solapa con el de la estadística inferencial, ya que las dos disciplinas se basan en el análisis de datos.

***Palabras clave*— Python, Machine Learning, patrones.**

***Abstract*— Machine Learning is a scientific discipline in the field of Artificial Intelligence that creates systems that learn automatically. Learning in this context means identifying complex patterns in millions of data. The machine that really learns is an algorithm that reviews the data and is able to predict future behavior. Automatically, also in this context, implies that these systems are improved autonomously over time, without human intervention.**

On many occasions the field of action of automatic learning overlaps with that of inferential statistics, since both disciplines are based on data analysis.

***Key Word* — Python, Machine Learning, patterns.**

I. INTRODUCCIÓN

El aprendizaje automático o aprendizaje automatizado o aprendizaje de máquinas (del inglés, machine learning) es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan. Se dice que un agente aprende cuando su desempeño mejora con la experiencia; es decir, cuando la habilidad no estaba presente en su genotipo o rasgos de nacimiento. De forma más concreta, los investigadores del aprendizaje de máquinas buscan algoritmos y heurísticas para convertir muestras de datos en programas de computadora, sin tener que escribir los últimos explícitamente. Los modelos o programas resultantes deben ser capaces de generalizar comportamientos e inferencias para un conjunto más amplio (potencialmente infinito) de datos.

II. CONTENIDO

```python
# Se importa la librería numpy
import numpy as np

# Se crea un vector (array) con seis elementos
a = np.array([0,1,2,3,4,5])

# Se imprime el array ... a
print(a, '\n')

# Número de dimensiones del array
print(a.ndim, '\n')

# Número de elementos del array
print(a.shape)
```

Figura 1.

Para iniciar a desarrollar procesos de machine Learning en Python se deben conocer las librerías pueden permitir realizar el proceso de una manera óptima.
La primera librería que se debe tener en cuenta es "***numpy***", la cual está enfocada en el manejo y uso de números, ya sea en grandes o pequeñas cantidades de los mismos.

En la figura 1 podemos observar que la librería "***numpy***" nos permite crear arreglos y matrices, de esta manera podemos manejar eficazmente grandes cantidades de datos, además de que facilita diferentes procesos.

```python
# Se cambia la estructura del array
b = a.reshape((3,2))
print(b, '\n')

# Se verifican los cambios
print(b.ndim, '\n')
print(b.shape)
```

Figura 2.

En la figura 2, vemos que se cambia la estructura del array a con la funcion reshape convirtiendose en una matriz con la misma cantidad de elementos creados anteriormente.
La nueva matriz posee dos tres filas y dos columnas.

```
# Se realiza una copia del array
c = a.reshape((3,2)).copy()
print(c, '\n')

# Se cambia el primer valor de c
c[0][0] = -99

# El array a no se modifica
print(a, '\n')

# El array c queda modificado
print(c)
```

Figura 3.

En la figura anterior podemos ver que la funcion copy realiza una copia del array, luego cambia el valor del primer elemento pero no para el vector a, solo en el c.

```
# Control de valores erróneos
c = np.array([1, 2, np.NAN, 3, 4])
print(c, 'n')

# Se verifica la existencia de valores nan
print(np.isnan(c), '\n')

# Se eligen todos los valores que NO son nan
print(c[~np.isnan(c)], '\n')

# Se calcula el promedio de los valores que NO son nan
print(np.mean(c[~np.isnan(c)]))
```

Figura 4.

Ahora vemos que se crea un nuevo vector para tener el control de valores erróneos, donde se encuentra un elemento llamado "***nan***" el cual es de valor inválido.

```
# Vamos a desarrollar un programa de machine learning (básico)
# El siguiente es un paquete de datos a ser procesdos:
# La primera columna es: Número de horas
# La segunda columna es: Número de tareas ejecutadas
data = np.genfromtxt("web_traffic.tsv", delimiter="\t")
print(data[:10], '\n')

# Número de datos
print(data.shape)
```

Figura 5.

Como se observa en la figura 5, cuando se enseña el contenido del documento de lectura, en este caso solo los 10 primeros elementos, se observa que los datos están distribuidos en filas, cada una con 2 valores, donde el primer valor equivale al tiempo (en horas) y el segundo al número de intentos de ingreso a la web X (en unidades).

```
# Se divide el array en dos vectores columnas: x, y
x = data[:,0]
y = data[:,1]

# Se muestran los valores en x, y
print(x, '\n')
print(y, '\n')
```

Figura 6.

El array se divide en dos vectores llamados "***x***" y "***y***".

```
# Dimensión de los vectores x, y
print(x.ndim, '\n')
print(y.ndim, '\n')

# Elementos contenidos en los vectores x, y
print(x.shape, '\n')
print(y.shape)
```

Figura 7.

Los vectores tienen la dimension de 1, ademas se verifica que la cantidad de datos sea la misma que se visualizo en un principio.

```
# Número de elementos en x, y, antes de ser comprimidos
print(x.shape, '\n')
print(y.shape, '\n')

# Se eliminan los elementos nan tanto de x como de y
x = x[~np.isnan(y)]
y = y[~np.isnan(y)]

# Se cuenta el número de elementos tanto de x como de y
print(x.shape, '\n')
print(x.shape, '\n')
```

Figura 8.

Luego de realizar procesos de clasificación y separación de los datos, se eliminan de los vectores "***x***" y "***y***" todos aquellos que son inválidos, los cuales no contribuyen con información al desarrollo del proceso.

```
# Se importa la librería para graficar
import matplotlib.pyplot as plt

# Dibuja los puntos (x,y) con círculos de tamaño 10
plt.scatter(x, y, s=10)

# Títulos de la gráfica
plt.title("Tráfico Web en el último mes")
plt.xlabel("Tiempo")
plt.ylabel("Accesos/Hora")


plt.xticks([w*7*24 for w in range(10)],['semana %i' % w for w in range(10)])
plt.autoscale(tight=True)

# dibuja una cuadrícula punteada ligeramente opaca
plt.grid(True, linestyle='-', color='0.75')

# Muestra el gráfico
plt.show()
```

Finalmente, se hace un proceso de graficado, usando la librería "***matplotib.pyplot***", en este caso se usarán las funciones que

permiten crear un plano coordenado para ubicar las parejas ordenadas de los datos ya mencionados.

III. RESULTADOS

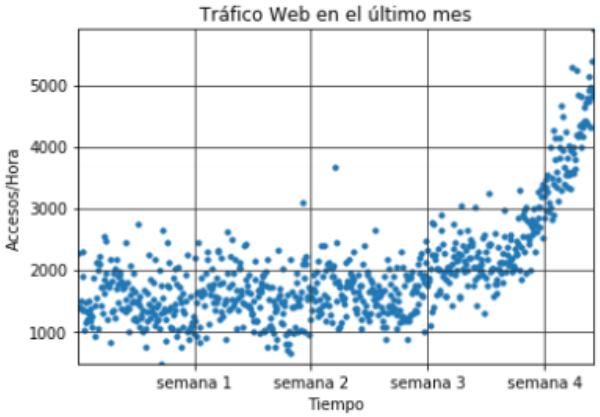

Figura 7.

IV. CONCLUSIONES

Dentro de los procesos del Machine Learning existen elementos que todavía cuentan con áreas de oportunidad. Sin embargo, también se han encontrado diversas alternativas para mitigar los posibles riesgos. Es importante tener en cuenta que mientras más robusto sea el diseño de datos, las posibilidades de que haya un modelo predictivo mal formulado son menores.

V. REFERENCIAS

https://matplotlib.org.
https://cleverdata.io/que-es-machine-learning-big-data/
https://numpy.org