PANDAS

Pandas es un paquete de manipulación de datos y análisis en Python. Es una biblioteca muy relacionada con numpy. Numpy se encarga de leer los datos (grandes cantidades de datos) y pandas se encarga de manipular estos datos, usando Series y Data Frames. Es similar a NumPy, que continene únicamente vectores, matrices o tensores junto con operaciones matemáticas. En Pandas, tenemos más cosas, como nombrar columnas, incluir índices o generar gráficas.
En IA, Pandas nos permite hacer un análisis previo de los datos y hacer que estos funcionen para nuestros algoritmos.

Instalación:

```
conda install pandas
```

Importación:

```
#Importing pandas with an Commonly used alias pd
import pandas as pd
```

Series:

```
#Crear una serie:
estudiantes = pd.Series(data=['Luis', 'Francia', 40], index=['nombre', 'pais', 'edad'])
estudiantes
print(estudiantes.shape)  #3 valores, una sola dimensión
print(estudiantes.ndim)
print(estudiantes.size)

print(estudiantes.values)  #la lista “data”
print(estudiantes.index)  #la lista “index”

print('nombre' in estudiantes)  #saber si un índice está dentro de la serie, ejemplo “compañía”.

#Acceder, Eliminar y modificar elementos de la serie:
#mediante el índice:
print(estudiantes['nombre'])  #El valor de nombre o de edad
print(estudiantes[['nombre', 'edad']])  #acceder a varios índices.
#mediante sus posiciones:
print(estudiantes[0])  #nombre o país [1]. También [[0,1]] nombre y país. O … acceder al último [-1], edad.
#función 'loc', que significa la ubicación del índice:
print (estudiantes.loc['pais'])  #función explícita, solo podemos agregar los índices que hemos declarado.
#función 'iloc', la ubicación numérica del índice, la posición.
print (estudiantes.iloc[0])
#'drop' elimina elementos de una serie:
estudiantes.drop(['edad'])  #lo elimina, pero NO MODIFICA LA SERIE
estudiantes.drop(['edad'], inplace = True)  #para que la modifique



#Operaciones aritméticas con Series:
import pandas as pd

fruits = pd.Series(data = [20, 40, 60], index = ['mango', 'manzana', 'platano'])
#(también se pueden crear sin llamar al nombre del argumento:
fruits = pd.Series([20, 40, 60],['mango', 'manzana', 'platano'])
```

```
#operación de suma, resta, multiplicación, división:
print(fruits + 3)  #se le suma a cada índice. Igual con la resta…
print(fruits/10)

#para usar combinada con numpy:
import numpy as np

print(np.sqrt(fruits))

print(np.power(fruits, 2))


#También sólo a un índice o varios:
print(fruits['mango'] + 3)
print(fruits[['platano', 'manzana']] * 10)
print(fruits.loc['manzana'] / 10
```

Crear DataFrames:

Objeto de dos dimensiones con filas y columnas etiquetadas. Cada fila y cada columna tienen un nombre.

```
#Crear un DataFrame con un diccionario:

items = {'Pepe': [10, 20, 30],
         'Ana': [40, 50, 60]}
df = pd.DataFrame(items)  #sin indices

items = {'Pepe': pd.Series([10, 20, 30], ['manzana', 'platano', 'mango']),
         'Ana': pd.Series([40, 50, 60], ['manzana', 'naranja', 'mango'])}
df = pd.DataFrame(items)  #NaN (Not a Number)

#una sóla columna y con los índices que quiero mostrar
df = pd.DataFrame(items, columns = ['Pepe'], index = ['manzana', 'platano'])


#Otro ejemplo (tambiém desde un diccionario):
df = pd.DataFrame({
    'temperatura':[35.5,36.7,22.3,39.7],
    'presion':[3.6,2.4,5.5,5.5]
},index=['3h','6h','9h','12h'])
```

#En pandas a los valores de la propiedad index se les llama etiquetas (label en inglés)

```
#desde un array:
df=pd.DataFrame(
    [[35.5,3.6], [36.7, 2.4], [22.3, 5.5], [39.7, 5.5]],
columns=['temperatura', 'presion'],
index=['3h','6h','9h','12h'])

#Añadir una nueva columna al inicio del DataFrame:
df.insert(0,"humedad",[65,63,65,67])

#Añadir una nueva columna, después de la columna "presion" ya que es la 2º del DataFrame
df.insert(2,"humedad",[65,63,65,67])
```

```
#Cargar los datos desde un fichero de texto llamado “datos.txt”, separados por una coma:
temperatura,presion,humedad
35.5,3.6,65
36.7,2.4,63
22.3,5.5,65
39.7,5.5,67
df=pd.read_csv("datos.txt",sep=",")


#otro ejemplo. Datos sacados de Kaggle. Kaggle es una plataforma de competencia para datascientists donde algunas compañías suben sus datos y plantean un problema. Las mejores soluciones se hacen acreedoras de un premio pero también esta plataforma es muy útil porque podemos encontrar datasets públicos y privados.

df = pd.read_csv('data_fifa.csv')


#Cargar los datos desde una base de datos relacional:
import sqlalchemy

connection=sqlalchemy.create_engine('mysql+pymysql://
mi_usuario:mi_contrasenya@localhost:3306/mi_database')
df=pd.read_sql("SELECT * FROM mi_tabla",con=connection)
#previamente hay que instalar sqlalchemy con:
conda install -c anaconda sqlalchemy
```

Mostrar tabla con datos:

```
#Mostrar las primeras filas
df.head()
df.head(10)

#Mostrar las últimas filas
df.tail()
df.tail(2)

#Mostrar las variables estadísticas más importantes de un datasets
df.describe()

#El máximo valor de acuerdo a las columnas
df.max()        #también df.min()

#Correlación, para saber cómo las columnas están relacionadas o no
df.corr()       #columna edad, con potencial, correlación negativa
```

Acceder a los datos:

```
#iloc(nº fila, nº columna) Accedemos por el número de fila y/o columna
#loc(etiqueta fila, nombre columna) Accedemos por etiqueta de la fila y/o nombre de la columnas

#El NÚMERO DE LA FILA O COLUMNA, PUEDE SER:
# el valor:  4
# un rango:  0:3
# un array con valores:  [0,4,5]
# todos:  :
```

```
#La ETIQUETA FILA, NOMBRE COLUMNA, PUEDEN SER:
# el valor:  "3h" o "temperatura"
# un rango:  "3h":"9h" o "humedad":"presion"
# un array con los valores:  ["3h","9h"] o ["humedad","presion"]
# todos:  :


#iloc
#Una celda por nº fila y nº columna
df.iloc[0,1]

#Una fila y cualquier columnas
df.iloc[0,:]
df.iloc[0]  #es lo mismo

#Cualquier número de fila y una columnas
df.iloc[:,1]

#Cualquier número de fila y cualquier de columna
df.iloc[:,:]
#loc
#Una celda por etiqueta y el nombre de columna
df.loc['3h','temperatura']

#Una fila por etiqueta de la fila y cualquier nombre de columna
df.loc['3h',:]
df.loc['3h']

#Cualquier etiqueta de fila y el nombre de columnas
df.loc[:,'temperatura']

#UNA FORMA ALTERNATIVA ES USAR:
df["temperatura"]

#Cualquier etiqueta de fila y nombre de columna
df.loc[:,:]
#ACCEDER A DATOS INDIVIDUALES MEZCLANDO ÍNDICES, ETIQUETAS Y
NOMBRES

#por columna
##nombre de columna y número de fila
df.loc[:,'temperatura'][0]

##nº columna y etiqueta
df.iloc[:,1]['3h']

#por fila
##etiqueta y nº columna
df.loc['3h'][1]

##nº fila y nombre de columna
df.iloc[0]['temperatura']


#ACCEDER A LOS NOMBRES DE LAS ETIQUETAS DEL DATAFRAME
#todas
df.index

#por número de fila
df.index[0]
```

#ACCEDER A LOS NOMBRES DE LAS COLUMNAS DEL DATAFRAME

```
#todas
df.columns

#nombre de columna por el nº de columna
df.columns[0]
```

#ACCEDER A LOS VALORES DEL DATAFRAME

```
df.values
```

#OTROS MÉTODOS

```
#obtener el número de datos a “null” de cada columna
df.isnull().sum()

#tipos de datos
df.dtypes
#características del DataFrame
df.info()

#dimensión de los datos (4 filas y 3 columnas)
df.shape

#número de filas
df.shape[0]
#número de columnas
df.shape[1]

#cuántos valores hay distintos
df.groupby('presion').size()

#consultas “query”
df.query('temperatura>36 and presion<3.0')

#cálculos “eval”
df.eval('temperatura*humedad')



#Volviendo al ejemplo FIFA:
#df.head(3)
#¿cuántos jugadores están en el FC Barcelona?

df[df['Club'] == 'FC Barcelona']        #33 jugadores


#Overall es la puntuación que se le da en FIFA

df[df['Overall'] == df['Overall'].max()]          #o min()


#¿Cuántos jugadores son de España?

df[df['Nationality'] == 'Spain']        #1072
```

Cambiar la estructura:

#el método **melt**
#Con los siguientes datos:

```
alumno,asignatura,eval1,eval2,eval3,final
Ana,Soci,4,7,7,6
Ana,Natu,5,7,8,7
Ana,Plasti,8,8,7,8
Ana,Física,8,9,9,9
Ana,Caste,7,7,6,7
Ana,Mate,5,5,6,5
Ana,Inglés,7,5,6,6
Ana,Valen,6,6,5,6
Ana,Valores,9,9,9,9
Sergio,Soci,5,8,6,6
Sergio,Natu,6,8,7,7
Sergio,Plasti,6,6,6,6
Sergio,Física,8,9,8,8
Sergio,Caste,7,7,7,7
Sergio,Mate,5,6,5,5
Sergio,Inglés,7,6,5,6
Sergio,Valen,6,5,5,5
Sergio,Valores,9,9,9,9
```

"""
Las notas es el resultado, pero hay 4 columnas de las notas (que son 1º Evaluacion, 2º Evaluacion, 3º Evaluacion y Final). Transformar con **melt** para que haya una única columna llamada *notas*(*value*) y que añada una nueva columna llamada *eval* con el tipo de nota, que es lo que llamaremos columna *var*
"""
"""
El método **melt** tiene los siguientes argumentos:

id_vars: Array con las columnas que no tienen los resultados, es decir, que no tiene las notas.
var_name: El nombre de la nueva columna que se va a crear con el tipo de nota. Es decir que dirá si es de la 1º Eval, 2º Eval, etc.
value_name: Nombre de la columna donde finalmente estará el resultado. Es decir, la nota.
"""

```
df= pd.read_csv("notas.txt",sep=",")
datos = df.melt(id_vars = ["alumno", "asignatura"], var_name = "eval", value_name = 'nota')

#Hemos transformado las columnas "eval1,eval2,eval3,final" es dos columnas llamadas "eval" y "nota". Siendo el contenido de la columna eval "eval1" , "eval2" , "eval3" o "final".
```

<--enlace método melt en pandas, más explicación-→

#Asimetría de los datos:
#Ayuda a saber si la distribución es normal

```
df.skew()    #humedad, temperatura y presión
```

#La función Pandas **dataframe.skew()** devuelve un sesgo imparcial sobre el eje solicitado Normalizado por N-1. La asimetría es una medida de la asimetría de la distribución de probabilidad de una variable aleatoria de valor real sobre su media. Para obtener más información sobre la asimetría, consulte este <-enlace->.

```
#otro ejemplo más amplio:
import pandas as pd
df = pd.read_csv("nba.csv")
df
df.skew(axis = 0, skipna = True)
```

```
#encontrar la asimetría de los datos sobre el eje de la columna:
import pandas as pd
df = pd.read_csv("nba.csv")
df.skew(axis = 1, skipna = True)
```

GRÁFICAS:

```
#Histograma de todas las columnas
df.hist()

#Función de densidad. (podemos ver la función que sigue)
df.plot(kind = 'density',
          subplots = True,
          layout = (3,3),
          figsize=(10,10),
          sharex = False)
```

PANDAS_PROFILING:

#**Genera HTML con información del DataFrames**

```
from pandas_profiling import ProfileReport
reporte = ProfileReport(df, title = "Mi reporte")
reporte.to_file("reporte.html")
```

<--enlace-->

##OTRA HERRAMIENTA SIMILAR PERO MÁS POTENTE ES **pywedge**
<-enlace1-->
<--enlace2-->