Investigación 1

Alvaro Salazar
Instituto Tecnológico de Costa Rica
San José, Costa Rica

***Resumen*—Investigación sobre el uso de Jupyter Notebook y Pandas basados en Python3. Se muestran ejemplos de operaciones básicas.**

Palabras claves—Jupyter, Notebook, Python, Pandas.

I. Introducción

La Investigación 1 está centrada en la comprensión del funcionamiento de *Jupyter Notebook* y Pandas basados en Python3, el cual es un lenguaje de programación muy popular actualmente.

El propósito de esta investigación es introducir al lector en la configuración del ambiente de trabajo para utilizar la biblioteca Pandas con la interfaz *Jupyter Notebook* con el lenguaje Python3. Se realizarán algunas operaciones básicas a modo de guía para ilustrar el funcionamiento.

II. Instalación

Se debe instalar la distribución Miniconda del sitio *https://conda.io/miniconda.html* y posteriormente utilizar los siguientes comandos en la terminal Anaconda:

```
conda create --name Inv-Corta-1 python=3
conda activate Inv-Corta-1
conda install numpy matplotlib pandas
scikit-learn jupyter notebook
```

III. Ejecución

A. Inicio de Jupyter Notebook

Se introduce el siguiente comando en la terminal para ejecutar *Jupyter Notebook*:

```
jupyter notebook
```

Con esto inicia un explorador de Internet con la interfaz de *Jupyter Notebook*. Se realiza *click* en *New* y después en *Python 3* para crear un nuevo archivo donde se realizarán los ejemplos demostrativos de Panda. La FIGURA 1 muestra la ubicación del botón *New* y la opción *Python 3*. La FIGURA 2 es un ejemplo de un nuevo archivo en *Jupyter Notebook* donde se realizarán ejemplos de Panda.

B. Creación de objetos

Inicialmente se debe importar librerías a utilizar. Opcionalmente se le ponen etiquetas.

```
In [1]: import numpy as np
In [2]: import pandas as pd
```

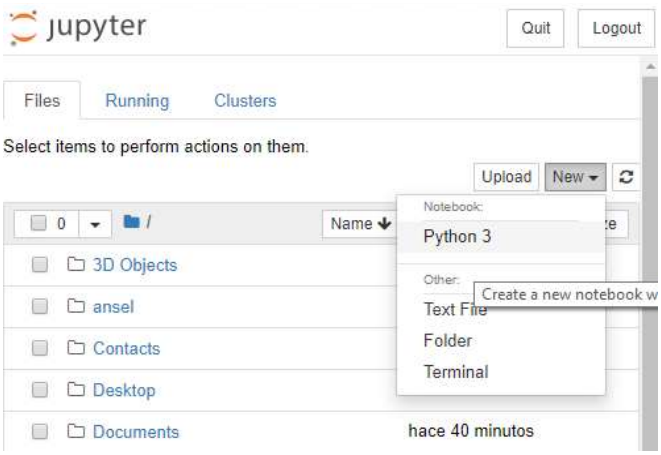

FIGURA 1 Ubicación del botón New en Jupyter Notebook.

El botón New permite crear nuevos archivos del ambiente de trabajo.

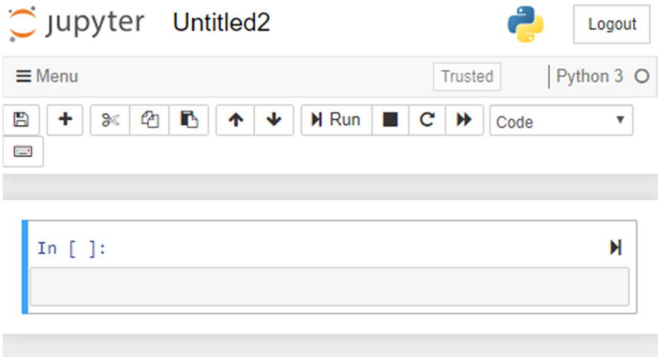

FIGURA 2 Nuevo archivo de *Jupyter Notebook*.

Nuevo archivo de *Jupyter Notebook* para escribir código.

Los siguientes códigos muestran ejemplos de creación de objetos. Note que el tipo de variables se ajustan automáticamente y cambian incluso entre columnas, como el caso de la línea 10.

```
In [3]: s = pd.Series([1, 3, 5, np.nan, 6, 8])
In [4]: s
Out[4]: 0    1.0
        1    3.0
        2    5.0
        3    NaN
        4    6.0
        5    8.0
        dtype: float64
```

```
In [5]: dates = pd.date_range('20130101', periods=6)

In [6]: dates
Out[6]: DatetimeIndex(['2013-01-01', '2013-01-02', '2013-01-03', '2013-01-04',
                       '2013-01-05', '2013-01-06'],
                      dtype='datetime64[ns]', freq='D')

In [7]: df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

In [8]: df
Out[8]:
```

	A	B	C	D
2013-01-01	-0.271982	-0.624141	0.306430	0.401306
2013-01-02	0.185927	0.023354	-0.718394	0.644060
2013-01-03	-0.072246	-0.166260	0.301814	0.357230
2013-01-04	1.001478	-0.316920	2.263932	0.347777
2013-01-05	0.399040	0.201875	0.030987	-0.122871
2013-01-06	1.146186	0.003106	1.016592	-0.211554

```
In [9]: df2 = pd.DataFrame({'A': 1.,
                           'B': pd.Timestamp('20130102'),
                           'C': pd.Series(1, index=list(range(4)), dtype='float32'
                           'D': np.array([3] * 4, dtype='int32'),
                           'E': pd.Categorical(["test", "train", "test", "train"]),
                           'F': 'foo'})

In [10]: df2
Out[10]:
```

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

```
In [11]: df2.dtypes
Out[11]: A           float64
         B    datetime64[ns]
         C           float32
         D             int32
         E          category
         F            object
         dtype: object
```

C. *Pre-procesamiento*

La TABLA I muestra varios comandos para el pre-procesamiento de datos. Algunos requieren argumentos como columnas, filas, constantes o modos. Se pone el nombre del conjunto de datos (como df que es nombre común de la guía de pandas), seguido de un punto y el comando. Por ejemplo: df.head(3) muestra los primeros 3 datos del conjunto df.

D. *Análisis exploratorio de datos*

El siguiente código permite crear una serie de datos para un ejemplo de exploración de datos.

```
a = pd.Series(np.random.randint(0, 10,
size=20))
b = pd.Series(np.random.randint(0, 10,
size=20))
c = pd.Series(np.random.randint(0, 10,
size=20))
s = pd.concat([a, b, c])
```

Teniendo *s* como el conjunto de datos a analizar, se utilizarán los siguientes comandos para comprender la distribución de los datos.

```
s.mean()
s.value_counts()
s.describe()
```

TABLA I COMANDOS DE PRE-PROCESAMIENTO DE DATOS

Comando	**Acción**
head(k)	Muestra primeros k datos. Si no se introduce k, k es 5 por defecto.
tail(k)	Muestra últimos k datos. Si no se introduce k, k es 5 por defecto.
index	Muestra la columna de índice.
columns	Muestra los encabezados de las columnas.
describe()	Muestra resumen estadístico de los datos por columna.
T	Transpone los datos.
df['A']	Muestra datos de la columna A.
df[a:b]	Muestra datos desde la fila a hasta la anterior a b.
df[c:d]	Muestra datos desde la fila con el encabezado c hasta la fila con el encabezado d.
df[df.A > 0]	Muestra datos de las filas con valores de columna A mayores a 0.
dropna(how='any')	Elimina filas con datos nulos.
fillna(value=k)	Sustituye datos nulos con el valor k.
mean()	Saca el promedio por columna.
mean(1)	Saca el promedio por fila.
value_counts()	Cuenta cada valor.
concat(a)	Concatena elementos que se mencionan en a.
merge(a, b, on='A')	Une como SQL a y b usando como clave a A.
apend(a, ignore_index=True)	Añade a al conjunto ignorando índices.
groupby(A).sum()	Suma por elemento de columna A.
pivot_table(A, values='B', index['C', 'D'], columns['E'])	Crea tabla pivote tomando los valores de columna B, índices C y D, columna E como encabezados, del conjunto A.
plot()	Dibuja una gráfica con los datos.

La FIGURA 3 muestra los resultados de los comandos recién mencionados. Se observa que los valores que más veces aparecen son 7, 9 y 3, que podrían considerarse como valores extremos, no centrales; por lo que se puede inferir que la distribución no es normal (gaussiana). Curiosamente, los valores 4 y 5 son los que menos veces aparecen.

Finalmente, la muestra la gráfica que se produce con el mismo conjunto utilizando el comando plot, lo cual muestra datos aleatorios.

IV. CONCLUSIONES

Jupyter Notebook demuestra ser una interfaz fácil de utilizar, así como pandas es una biblioteca con funciones sumamente útiles para el análisis y procesamiento de datos que en esta guía se utilizaron para pre-procesamiento.

```
In [32]: s.mean()
Out[32]: 4.916666666666667

In [33]: s.value_counts()
Out[33]: 7    10
         9     9
         3     8
         8     7
         2     7
         0     6
         6     4
         1     4
         4     3
         5     2
         dtype: int64

In [41]: s.describe()
Out[41]: count    60.000000
         mean      4.916667
         std       3.087930
         min       0.000000
         25%       2.000000
         50%       5.500000
         75%       8.000000
         max       9.000000
         dtype: float64
```

FIGURA 3 RESULTADOS DE COMANDOS EXPLORATORIOS.

Los resultados permiten inferir la distribución y tipo de datos.

```
In [43]: s.plot()
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x
         2034b958898>
```

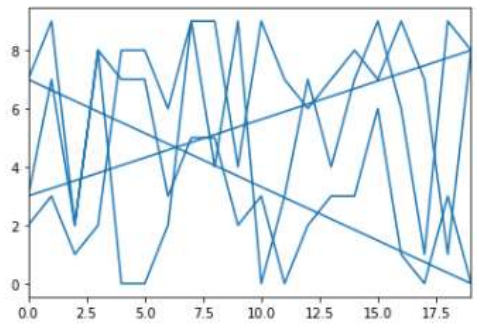

FIGURA 4 GRÁFICA DEL CONJUNTO DE DATOS.

Los datos son totalmente aleatorios.

Se adjunta esta guía junto al archivo *Jupyter Notebook* para ser enviado al profesor del curso "Reconocimiento de Patrones" del Instituto Tecnológico de Costa Rica.

V. BIBLIOGRAFÍA

"Ambiente de Trabajo *Python*." Meza, F. Escuela de Electrónica. Instituto Tecnológico de Costa Rica.

"*10 Minutes to pandas*". Pandas.pydata.org. 5/6/2019. http://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#operations