GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN

CIENCIA DE DATOS

Agenda
Sesión 9/18

Numpy

- ¿Qué es Pandas?
- Series
- DataFrame
- Lectura de Archivos
- Limpieza de Datos
- Corelaciones

¿Qué es Pandas?

Pandas es una biblioteca de Python.

Pandas se utiliza para analizar datos.

¿Qué son los pandas?

Pandas es una biblioteca de Python que se usa para trabajar con conjuntos de datos.

Tiene funciones para analizar, limpiar, explorar y manipular datos.

El nombre "Pandas" hace referencia tanto a "Datos de panel" como a "Análisis de datos de Python" y fue creado por Wes McKinney en 2008.

¿Qué es Pandas?

¿Por qué utilizar Pandas?

Pandas nos permite analizar big data y sacar conclusiones basadas en teorías estadísticas.

Los pandas pueden limpiar conjuntos de datos desordenados y hacerlos legibles y relevantes.

Los datos relevantes son muy importantes en la ciencia de datos.

¿Qué es Pandas?

¿Qué pueden hacer los pandas?

Pandas te da respuestas sobre los datos como:

¿Existe una correlación entre dos o más columnas?

¿Qué es el valor medio?

¿Valor máximo?

¿Valor mínimo?

Los pandas también pueden eliminar filas que no son relevantes o que contienen valores incorrectos, como valores vacíos o NULL. A esto se le llama limpiar los datos.

¿Qué es Pandas?

¿Dónde está la base de código de Pandas?

El código fuente de Pandas se encuentra en este repositorio de github

https://github.com/pandas-dev/pandas

¿Qué es Pandas?

Instalación de Pandas

Si ya tiene Python y PIP instalados en un sistema, la instalación de Pandas es muy fácil.

Instálelo usando este comando:

pip install pandas

Si este comando falla, use una distribución de Python que ya tenga Pandas instalado como, Anaconda, Spyder, etc.

¿Qué es Pandas?

Importar pandas

Una vez que Pandas esté instalado, impórtelo en sus aplicaciones agregando la palabra import clave:

```
import pandas
```

¿Qué es Pandas?

Ahora Pandas está importado y listo para usar.

```
import pandas


mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}
myvar = pandas.DataFrame(mydataset)
print(myvar)
```

Series

¿Qué es una serie?

Una serie Pandas es como una columna en una tabla.

Es una matriz unidimensional que contiene datos de cualquier tipo.

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a)
print(myvar)
```

Series

Crear etiquetas

Con el argumento index, puede nombrar sus propias etiquetas.

```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

Series

Cuando haya creado etiquetas, puede acceder a un elemento consultando la etiqueta.

Devuelve el valor de "y":

print(myvar["y"])

Series

Objetos clave / valor como serie

También puede utilizar un objeto clave / valor, como un diccionario, al crear una serie.

```
import pandas as pd
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories)
print(myvar)
```

DataFrame

¿Qué es un DataFrame?

Un DataFrame de Pandas es una estructura de datos bidimensional, como una matriz bidimensional o una tabla con filas y columnas.

```
import pandas as pd
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
```

DataFrame

Localizar fila

Como puede ver en el resultado anterior, el DataFrame es como una tabla con filas y columnas.

Los pandas usan el atributo loc para devolver una o más filas especificadas

```
#refer to the row index:
print(df.loc[0])
```

Lectura de Archivos - CSV

Leer archivos CSV

Una forma sencilla de almacenar grandes conjuntos de datos es utilizar archivos CSV (archivos separados por comas).

Los archivos CSV contienen texto sin formato y es un formato bien conocido que puede ser leído por todos, incluidos Pandas.

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.to_string())
```

Lectura de Archivos - CSV

De forma predeterminada, cuando imprime un DataFrame, solo obtendrá las primeras 5 filas y las últimas 5 filas:

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df)
```

Lectura de Archivos - JSON

Leer JSON

Los grandes conjuntos de datos a menudo se almacenan o se extraen como JSON.

JSON es texto sin formato, pero tiene el formato de un objeto y es muy conocido en el mundo de la programación, incluido Pandas.

```
import pandas as pd
df = pd.read_json('data.json')
print(df.to_string())
```

Pandas: análisis de marcos de datos

Ver los datos

Uno de los métodos más utilizados para obtener una descripción general rápida del DataFrame es el método head().

El método head() devuelve los encabezados y un número específico de filas, comenzando desde la parte superior.

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.head(10))
```

Pandas: análisis de marcos de datos

También hay un método tail() para ver las últimas filas del DataFrame.

El método tail() devuelve los encabezados y un número específico de filas, comenzando desde abajo.

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.tail(10))
```

Pandas: análisis de marcos de datos

Información sobre los datos

El objeto DataFrames tiene un método llamado info(), que le brinda más información sobre el conjunto de datos.

```
import pandas as pd

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

print(df.info())

# Imprimir información sobre los datos
```

Pandas: análisis de marcos de datos

Explicación de resultados

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  169 non-null    int64
 1   Pulse     169 non-null    int64
 2   Maxpulse  169 non-null    int64
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None
```

Pandas - Limpieza de Datos

Limpieza de datos

La limpieza de datos significa corregir los datos incorrectos en su conjunto de datos.

Los datos incorrectos pueden ser:

- Celdas vacías
- Datos en formato incorrecto
- Información incorrecta
- Duplicados

Pandas - Limpieza de Datos

Celdas vacías

Las celdas vacías pueden potencialmente darle un resultado incorrecto cuando analiza datos.

Quitar filas

Una forma de lidiar con las celdas vacías es eliminar las filas que contienen celdas vacías.

Esto suele estar bien, ya que los conjuntos de datos pueden ser muy grandes y eliminar algunas filas no tendrá un gran impacto en el resultado.

Pandas - Limpieza de Datos

```
import pandas as pd
df = pd.read_csv('dirtydata.csv')
new_df = df.dropna()
print(new_df.to_string())
```

Si desea cambiar el DataFrame original, use el argumento inplace = True:

```
import pandas as pd
df = pd.read_csv('dirtydata.csv')
df.dropna(inplace = True)
print(df.to_string())
```

Pandas - Limpieza de Datos

Reemplazar valores vacíos

Otra forma de lidiar con celdas vacías es insertar un nuevo valor en su lugar.

De esta manera, no tiene que eliminar filas enteras solo por algunas celdas vacías.

El método fillna() nos permite reemplazar celdas vacías con un valor:

```
import pandas as pd
df = pd.read_csv('dirtydata.csv')
df.fillna(130, inplace = True)
```

Pandas - Limpieza de Datos

Reemplazar solo para columnas especificadas

El ejemplo anterior reemplaza todas las celdas vacías en todo el marco de datos.

Para reemplazar solo los valores vacíos de una columna, especifique el nombre de la columna para el DataFrame:

```
import pandas as pd
df = pd.read_csv('dirtydata.csv')
df["Calories"].fillna(130, inplace = True)
```

Pandas - Limpieza de Datos

Calcule la MEDIA y reemplace los valores vacíos con ella:

```
import pandas as pd


df = pd.read_csv('data.csv')
x = df["Calories"].mean()
df["Calories"].fillna(x, inplace = True)
```

Media = el valor promedio (la suma de todos los valores dividida por el número de valores).

Pandas - Limpieza de Datos

Calcule la MEDIANA y reemplace los valores vacíos con ella:

```
import pandas as pd


df = pd.read_csv('data.csv')
x = df["Calories"].median()
df["Calories"].fillna(x, inplace = True)
```

Mediana = el valor en el medio, después de haber ordenado todos los valores en forma ascendente.

Pandas - Limpieza de Datos

Calcule la MODA y reemplace los valores vacíos con ella:

```
import pandas as pd

df = pd.read_csv('data.csv')
x = df["Calories"].mode()[0]
df["Calories"].fillna(x, inplace = True)
```

#**Mode** = el valor que aparece con mayor frecuencia.

Pandas - Limpieza de Datos

Calcule la MEDIA y reemplace los valores vacíos con ella:

```
import pandas as pd


df = pd.read_csv('data.csv')
x = df["Calories"].median()
df["Calories"].fillna(x, inplace = True)
```

Mediana = el valor en el medio, después de haber ordenado todos los valores en forma ascendente.

Pandas - Limpieza de Datos

Limpieza de Datos de formato incorrecto

Las celdas con datos de formato incorrecto pueden dificultar, o incluso imposibilitar, el análisis de datos.

Para solucionarlo, tiene dos opciones: eliminar las filas o convertir todas las celdas de las columnas en el mismo formato.

```
import pandas as pd
df = pd.read_csv('data.csv')
df['Date'] = pd.to_datetime(df['Date'])
print(df.to_string())

#Elimine filas con un valor NULL en la columna "Fecha":
df.dropna(subset=['Date'], inplace = True)
```

Pandas - Limpieza de Datos

Encontrar relaciones

Un gran aspecto del módulo Pandas es el método corr().

El método corr() calcula la relación entre cada columna de su conjunto de datos.

Muestre la relación entre las columnas:

```
df.corr()
```

Nota: El método corr() ignora las columnas "no numéricas".

Pandas - Limpieza de Datos

Resultado explicado

El resultado del corr()método es una tabla con muchos números que representa qué tan bien está la relación entre dos columnas.

El número varía de -1 a 1.

1 significa que hay una relación de 1 a 1 (una correlación perfecta), y para este conjunto de datos, cada vez que un valor subió en la primera columna, la otra subió también.

0.9 también es una buena relación, y si aumenta un valor, el otro probablemente también lo hará.

Pandas - Limpieza de Datos

Resultado explicado

-0,9 sería una relación tan buena como 0,9, pero si aumenta un valor, el otro probablemente bajará.

0.2 significa NO una buena relación, lo que significa que si un valor aumenta no significa que el otro lo hará.

¿Qué es una buena correlación? Depende del uso, pero creo que es seguro decir que debe tener al menos 0.6(o -0.6) para llamarlo una buena correlación.

Pandas - Limpieza de Datos

Correlación perfecta:

Podemos ver que "Duración" y "Duración" obtuvieron el número 1.000000, lo cual tiene sentido, cada columna siempre tiene una relación perfecta consigo misma.

Buena correlación:

"Duración" y "Calorías" tienen una correlación 0.922721, que es una muy buena correlación, y podemos predecir que cuanto más te ejercitas, más calorías quemas, y al revés: si quemas muchas calorías, Probablemente haya tenido un largo ejercicio.

Pandas - Limpieza de Datos

Mala correlación:

"Duración" y "Maxpulse" obtuvieron una correlación 0.009403, que es una muy mala correlación, lo que significa que no podemos predecir el pulso máximo simplemente mirando la duración del ejercicio y viceversa.

Numpy a Pandas

Convertir arrays de la líbrería de Numpy a Dataframes puede ser muy util

```
import numpy as np
import pandas as pd

# Creating the array to convert
numpy_array = np.array([[1, 'yo'], [4, 'bro'], [4, 'low'], [1, 'NumPy']])

# Create the dataframe
df = pd.DataFrame(numpy_array, columns=['digits', 'words'])

print(df)
```

RESUMEN DE CLASE

GOBIERNO DE
EL SALVADOR

SECRETARÍA DE
INNOVACIÓN