Python para Data Science

Machine Learning y Pronósticos de Regresión

Introducción

- El **Data Science** -Ciencia de Datos-se encarga de estudiar de dónde viene la información, qué representa y cómo se puede convertir en un recurso valioso en la creación de negocios y estrategias.
- Para ello, busca extraer grandes cantidades de datos para identificar patrones para ayudar a una organización a controlar los costos, aumentar la eficiencia, reconocer nuevas oportunidades de mercado y aumentar la ventaja competitiva de la organización.
- El Data Science emplea las disciplinas de las matemáticas, estadística y las ciencias de la computación. Además, se incorporación de técnicas como el **Machine Learning** -Aprendizaje Automático-, cluster análisis -análisis de grupos-, Data Mining -extracción de datos-

Introducción

- El **Machine Learning** –aprendizaje automático– es una rama de la inteligencia artificial que permite que las máquinas aprendan sin ser expresamente programadas para ello, una habilidad indispensable para hacer sistemas capaces de identificar patrones entre los datos para hacer predicciones.
- Se usa en recomendaciones de películas en plataformas digitales de habla de asistentes virtuales, automóviles autónomos, motores de búsqueda, robótica, diagnóstico médico, detección de fraudes en pagos de tarjeta de crédito, etc.
- La estadística es la base fundamental del **Machine Learning**!

Cuales lenguajes se utilizan en Machine Learning?

- Python
- R
- MatLab
- Julia

- Lidera en lenguajes de desarrollo de Machine Learning debido a si simplicidad y facilidad de aprendizaje
- Es un éxito entre los principiantes que son nuevos en Machine Learning
- Viene con librerías específicas como NumPy, Pandas, SciKit-Learn que permite a la computadora aprender algebra lineal, equaciones diferenciales, regresión y otros métodos.

Python Libraries para Data Science

Librerías más populares en Python:

- NumPy — **Link:** http://www.numpy.org/
- SciPy — **Link:** https://www.scipy.org/scipylib/
- Pandas — **Link:** http://pandas.pydata.org/
- SciKit-Learn — **Link:** http://scikit-learn.org/

Librerías para visualización

- matplotlib — **Link:** https://matplotlib.org/
- Seaborn — **Link:** https://seaborn.pydata.org/

Inicio de Jupyter nootebook

```
# On the Shared Computing Cluster
[scc1 ~] jupyter notebook
```

Cargar librerías de Python

```
# On the Shared Computing Cluster
[scc1 ~] jupyter notebook
```

```
In [ ]: #Import Python Libraries
        import numpy as np
        import scipy as sp
        import pandas as pd
        import matplotlib as mpl
        import seaborn as sns
```

Presione `Shift+Enter` para ejecutar las celdas *jupyter*

Cargar librerías de Python

```
# On the Shared Computing Cluster
[scc1 ~] jupyter notebook
```

```
In [ ]: #Import Python Libraries
        import numpy as np
        import scipy as sp
        import pandas as pd
        import matplotlib as mpl
        import seaborn as sns
```

Presione `Shift+Enter` para ejecutar las celdas *jupyter*

Leer datos usando pandas

```
In [ ]: #Read csv file
        df = pd.read_csv("http://rcs.bu.edu/examples/python/data_analysis/Salaries.csv")
```

Otros formatos de documentos que se pueden leer con pandas

```
pd.read_excel('myfile.xlsx',sheet_name='Sheet1', index_col=None, na_values=['NA'])
pd.read_stata('myfile.dta')
pd.read_sas('myfile.sas7bdat')
pd.read_hdf('myfile.h5','df')
```

Explorando data frames

In [3]:
```
#List first 5 records
df.head()
```

Out[3]:

	rank	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
1	Prof	A	12	6	Male	93000
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205
4	Prof	B	20	18	Male	104800

Tipos de datos en Data Frame

Pandas Type	Native Python Type	Description
object	string	Mas general, puede tener mezclas de tipos (numbers and strings).
int64	int	Enteros
float64	float	Numéricos
datetime64, timedelta[ns]	N/A	Datos de tiempo

Tipos de datos en el Data Frame

```
In [4]: #Check una columna
        df['salary'].dtype
```

```
Out[4]: dtype('int64')
```

```
In [5]: #Check todas las columnas
        df.dtypes
```

```
Out[4]: rank          object
        discipline    object
        phd           int64
        service       int64
        sex           object
        salary        int64
        dtype: object
```

Atributos del Data Frames

Python objects have *attributes* and *methods*.

df.attribute	description
dtypes	lista de types en las columnas
columns	lista de nombres en las columnas
axes	Lista de las etiquetas en las filas y los nombres de la columnas
ndim	Número de dimensiones
size	Número de elementos
shape	Devuelve el par ordenado de los datos
values	Representación de datos en numpy

Métodos en Data Frames

Unlike attributes, python methods have *parenthesis.*
All attributes and methods can be listed with a *dir()* function: `dir(df)`

df.method()	descripción
head([n]), tail([n])	Primeras/últimas n columnas
describe()	Genera estadística descriptiva
max(), min()	Devuelve máximo / mínimo
mean(), median()	Devuelve media / mediana
std()	Desviación estándar
sample([n])	Muestra de n filas del data frame
dropna()	Elimina las entradas con datos faltantes

Seleccionar una columna en un Data Frame

Metodo 1: Selecionar usando el nombre de la columna:

df[“sex”]

Metodo 2: Use le nombre de la columna como atributo:

df.sex

Método *groupby* Data Frames

Usando el método "group by" podemos:

- Dividir los datos en grupos basados en algún criterio
- Calcular estadisticas (o aplicar una función) a cada grupo

```
In [ ]: df_rank = df.groupby(["rank"])
```

```
In [ ]: df_rank.mean()
```

rank	phd	service	salary
AssocProf	15.076923	11.307692	91786.230769
AsstProf	5.052632	2.210526	81362.789474
Prof	27.065217	21.413043	123624.804348

Método *groupby en* Data Frames

Una vez creado un objeto con groupby nosotros podemos calcular estadísticas para el grupo

In [ ]:
```
df.groupby("rank")[["salary"]].mean()
```

	salary
rank	
AssocProf	91786.230769
AsstProf	81362.789474
Prof	123624.804348

Filtrando Data Frame

Para dividir los datos en subconjuntos, podemos aplicar la indexación booleana. Esta indexación se conoce comúnmente como filtro. Por ejemplo, si queremos crear un subconjunto de las filas en las que el valor del salario es superior a $ 120K

```
In [ ]: df_sub = df[ df["salary"] > 120000 ]
```

Se puede usar cualquier operador booleano para dividir los datos en subconjuntos:

- mayor; >= mayor o igual;
- < menos; <= menor o igual;
- == igual; != no igual;

```
In [ ]: df_f = df[ df["sex"] == "Female" ]
```

Slicing en Data Frames

Hay varias formas de crear subconjuntos del marco de datos:
una o más columnas
una o más filas
un subconjunto de filas y columnas

Las filas y columnas se pueden seleccionar por su posición o etiqueta

Slicing en Data Frames

Al seleccionar una columna, es posible usar un solo conjunto de corchetes, pero el objeto resultante será una Serie (no es un DataFrame):

```
In [ ]: df["salary"]
```

Cuando necesitamos seleccionar más de una columna y/o hacer que la salida sea un DataFrame, debemos usar corchetes dobles:

```
In [ ]: df[["rank","salary"]]
```

Seleccionar filas en Data Frames

Si necesitamos seleccionar un rango de filas, podemos especificar el rango usando ":"

```
In [ ]: df[10:20]
```

Observe que la primera fila tiene una posición 0 y se omite el último valor del rango:
Entonces, para el rango 0:10, las primeras 10 filas se devuelven con las posiciones que comienzan con 0 y terminan con 9

Data Frames: method loc

If we need to select a range of rows, using their labels we can use method loc:

In [ ]:
```
#Select rows by their labels:
df_sub.loc[10:20,['rank','sex','salary']]
```

Out[ ]:

	rank	sex	salary
10	Prof	Male	128250
11	Prof	Male	134778
13	Prof	Male	162200
14	Prof	Male	153750
15	Prof	Male	150480
19	Prof	Male	150500

Método iloc en Data Frames:

Si necesitamos seleccionar un rango de filas y/o columnas, usando sus posiciones podemos usar el método iloc:

```
In [ ]: df_sub.iloc[10:20,[0, 3, 4, 5]]
```

Out[ ]:

	rank	service	sex	salary
26	Prof	19	Male	148750
27	Prof	43	Male	155865
29	Prof	20	Male	123683
31	Prof	21	Male	155750
35	Prof	23	Male	126933
36	Prof	45	Male	146856
39	Prof	18	Female	129000
40	Prof	36	Female	137000
44	Prof	19	Female	151768
45	Prof	25	Female	140096

Data Frames: Método iloc (resumen)

```
df.iloc[0]  # Primera fila en el data frames
df.iloc[i]  #(i+1)-enesima fila
df.iloc[-1] # Última fila
```

```
df.iloc[:, 0]  # Primera Columna
df.iloc[:, -1] # Última columna
```

```
df.iloc[0:7]        #Primeras 7 filas
df.iloc[:, 0:2]     #Primeras 2 columnas
df.iloc[1:3, 0:2]   #Segunda a tercera fila and primeras 2 columnas
df.iloc[[0,5], [1,3]]  #1ra y 6ta filas y 2da y 4ta columnas
```

Ordenar (Sorting) Data Frames

Podemos ordenar los datos por un valor en la columna. De forma predeterminada, la clasificación se realizará en orden ascendente y se devolverá un nuevo marco de datos.

In [ ]:
```
df_sorted = df.sort_values( by ='service')
df_sorted.head()
```

Out[ ]:

	rank	discipline	phd	service	sex	salary
55	AsstProf	A	2	0	Female	72500
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000

Datos faltantes

Dados faltantes se etiquetan con NaN

In [ ]:
```
flights[flights.isnull().any(axis=1)].head()
```

Out[ ]:

	year	month	day	dep_time	dep_delay	arr_time	arr_delay	carrier	tailnum	flight	origin	dest	air_time	distance	hour	minute
330	2013	1	1	1807.0	29.0	2251.0	NaN	UA	N31412	1228	EWR	SAN	NaN	2425	18.0	7.0
403	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EHAA	791	LGA	DFW	NaN	1389	NaN	NaN
404	2013	1	1	NaN	NaN	NaN	NaN	AA	N3EVAA	1925	LGA	MIA	NaN	1096	NaN	NaN
855	2013	1	2	2145.0	16.0	NaN	NaN	UA	N12221	1299	EWR	RSW	NaN	1068	21.0	45.0
858	2013	1	2	NaN	NaN	NaN	NaN	AA	NaN	133	JFK	LAX	NaN	2475	NaN	NaN

Datos Faltantes

Hay una serie de métodos para tratar los valores faltantes en el Data Frame:

df.method()	description
dropna()	Eliminar observaciones faltantes
dropna(how='all')	Eliminar observaciones donde todas las celdas son NA
dropna(axis=1, how='all')	Descartar columna si faltan todos los valores
dropna(thresh = 5)	Soltar filas que contienen menos de 5 valores no perdidos
fillna(0)	Reemplazar los valores faltantes con ceros
isnull()	devuelve True si falta el valor
notnull()	Devuelve True para valores que no faltan

Datos Faltantes

- Al sumar los datos, los valores faltantes se tratarán como cero
- Si faltan todos los valores, la suma será igual a NaN
- Los métodos cumsum() y cumprod() ignoran los valores faltantes pero los conservan en las matrices resultantes
- Se excluyen los valores faltantes en el método groupby
- Muchos métodos de estadísticas descriptivas tienen la opción skipna para controlar si se deben excluir los datos faltantes. Este valor se establece en True de forma predeterminada

Aggregation Functions in Pandas

Aggregation - computing a summary statistic about each group, i.e.

- compute group sums or means
- compute group sizes/counts

Common aggregation functions:

min, max
count, sum, prod
mean, median, mode, mad
std, var

Funciones de agregación en Pandas

El método agg() es útil cuando se calculan múltiples estadísticas por columna:

```
In [ ]: df[["service","salary"]].agg(['min','mean','max'])
```

Out[ ]:

	service	salary
min	0.000000	57800.000000
mean	15.051282	108023.782051
max	51.000000	186960.000000

Estadísticas Descriptivas Básicas

df.method()	descripción
describe	Estadísticas básicas (recuento, media, estándar, mínimo, cuantiles, máximo)
min, max	Valores mínimos y máximos
mean, median, mode	Media aritmética, mediana y moda
var, std	Varianza y desviación estándar
sem	Error estándar de la media
skew	asimetría de la muestra
kurt	kurtosis

Gráficos para datos.

El paquete Seaborn se basa en matplotlib pero proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos, similar a la biblioteca ggplot2 en R. Se dirige específicamente a la visualización de datos estadísticos

Para mostrar gráficos dentro del cuaderno de Python, incluya la directiva en línea:

```
In [ ]: %matplotlib inline
```

Gráficos

	descripción
distplot	Histograma
barplot	Estimación de tendencia central para una variable numérica
violinplot	Similar al diagrama de caja, también muestra la densidad de probabilidad de los datos
jointplot	Gráfico de dispersión
regplot	Gráfica de regresión
pairplot	Diagrama de pares
boxplot	Diagrama de caja
swarmplot	Diagrama de dispersión categórica
factorplot	Trama categórica general

Análisis estadístico básico

statsmodel y scikit-learn: ambos tienen una serie de funciones para el análisis estadístico

El primero se usa principalmente para el análisis regular usando fórmulas de estilo R, mientras que scikit-learn está más diseñado para Machine Learning.

statsmodels:

- Regresión Lineal
- ANOVA
- Pruebas de Hipótesis
- Muchos más ...

scikit-learn:

- Regresión Lineal
- kmeans
- máquinas de vectores de soporte
- bosques aleatorios
- Muchos más ...

Regresión Lineal Simple (SLR)

Podemos implementar SLR en Python de dos maneras, una es proporcionar su propio conjunto de datos y la otra es usar un conjunto de datos de la biblioteca scikit-learn python

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

In [ ]:
```
X = df.service[:, np.newaxis]
```

In [ ]:
```
X_train = X[:-30]
X_test = X[-30:]
```

In [ ]:
```
y_train = df.salary[:-30]
y_test = df.salary[-30:]
```

Regresión Lineal Simple (SLR)

Continuación

In [ ]:
```
from sklearn.linear_model import Ridge
```

In [ ]:
```
regr.fit(X_train, y_train)
```

In [ ]:
```
y_pred = regr.predict(X_test)
```

In [ ]:
```
print('Coefficients: \n', regr.coef_)
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

In [ ]:
```
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.show()
```

Regresión Lineal Simple (SLR)

Salidas

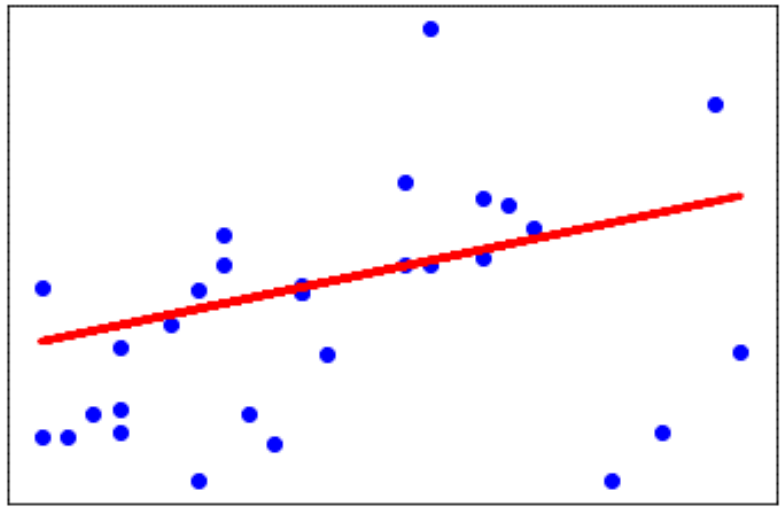
In [ ]:
```
print('Coefficients: \n', regr.coef_)
print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
print('Variance score: %.2f' % r2_score(y_test, y_pred))
```

```
Coefficients:
 [1166.69163414]
Mean squared error: 531411503.42
Variance score: 0.06
```

Regresión Lineal Simple (SLR)

Salidas

In [ ]:
```
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=3)
plt.xticks(())
plt.yticks(())
plt.show()
```



Proyecto de Investigación Pronóstico del Clima Base de datos NOAA (EE.UU)

- Trabajo Escrito (25%)
  - Portada
  - Resumen
  - Índice
  - Introducción
  - Marco Teórico
  - Desarrollo herramienta Machine Learning
  - Conclusiones
  - Bibliografía

- Programa (75%)
- Descarga de Datos 1 estación meteorológica del estado correspondiente.
  - Al menos 10 años
- Lectura de datos en Pandas file.csv
- Preparación de data para Machine Learning
- Entrenamiento del modelo (80% de los datos)
- Evaluación del modelo
- Creación de una función para hacer proyecciones