Práctica 1. Lenguajes para ciencia de datos: python

- Ejemplos de uso de librerías para ciencia de datos

NumPy

- NumPy: arrays multidimensionales, más un gran conjunto de operaciones matemáticas sobre esos arrays. Los arrays son bloques de datos organizados en múltiples dimensiones, que implementan los vectores y las matrices matemáticas. Las operaciones sobre matrices son rápidas y susceptibles de ser vectorizadas.

- **Website**: http://www.numpy.org/

- **import numpy as np**

SciPy

- SciPy complementa a NumPy y proporciona algoritmos para álgebra lineal, matrices dispersas, procesamiento de señales y de imágenes, optimización, análisis espectral (transformada rápida de Fourier), etc.

- **Website**: http://www.scipy.org/

Pandas

- Proporciona estructuras de datos para manejar tablas y series, no disponibles en NumPy/SciPy. Las estructuras DataFrame y Series permiten manejar tablas cuyas columnas tengan diferentes tipos, con operaciones para extraer subtablas, manejar elementos perdidos, agregar, cambiar de forma y visualizar la información.

- **Website**: http://pandas.pydata.org/

Scikit-learn

- Es una evolución de un toolkit de SciKit, y actualmente constituye el núcleo de las herramientas de ciencia de datos en Python. Contiene módulos de preprocesamiento de datos, aprendizaje supervisado y no supervisado, selección de modelos, validación, métricas de error, etc.

- **Website**: http://scikit-learn.org/stable/

Matplotlib

- Librería que contiene herramientas para crear gráficos a partir de arrays y visualizarlos interactivamente.
- Frameworks similares a Matlab dentro del módulo pylab
- **Website**: http://matplotlib.org/
- **import matplotlib.pyplot as plt**

Parte 1: scipy

- En data mining, Scipy se usa para tareas de estadística descriptiva y modelos estadísticos.
- Estudiamos algunos casos de uso:
  - Distribuciones de probabilidad (archivos scipy-1.py, scipy-2.py)
  - Tests estadísticos (archivos scipy-3.py, scipy-4.py)
  - Distancias entre instancias (archivo scipy-5.py)

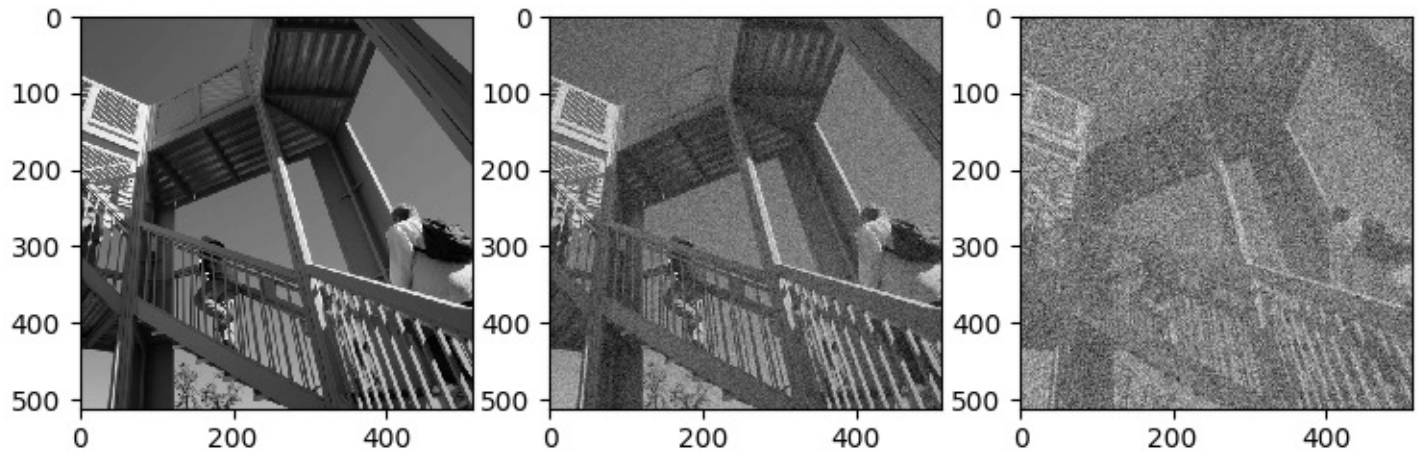
scipy-1

- Adición de ruido a una imagen

```
import scipy
import matplotlib.pyplot as plt
from scipy.stats import norm
face = scipy.misc.ascent().astype(float)
faceruido = face +
norm.rvs(loc=0,scale=16,size=face.shape)
plt.subplot(121,aspect='equal')
plt.imshow(face)
plt.gray()
plt.subplot(122,aspect='equal')
plt.imshow(faceruido)
plt.show()
```

Ejercicio 1

- Añade una tercera imagen al gráfico con un ruido de desviación típica 64. Debes obtener el resultado que se muestra en esta transparencia.



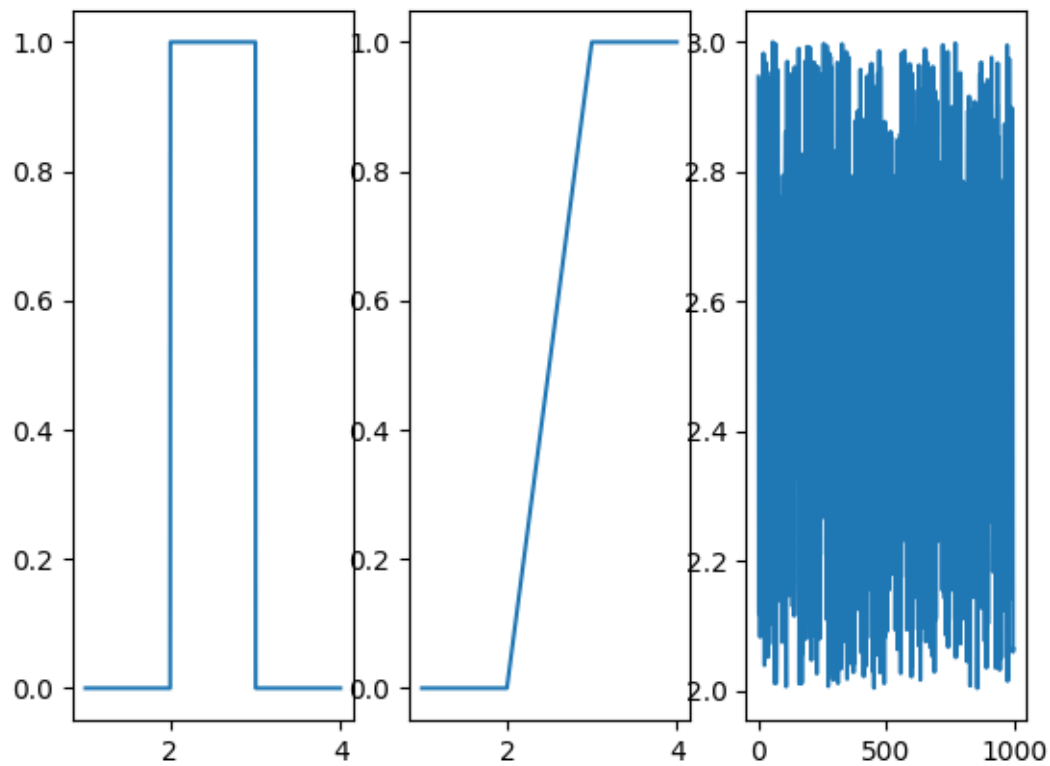
scipy-2

- Distribuciones de probabilidad

```
# Pareto
import numpy
import matplotlib.pyplot as plt
from scipy.stats import pareto
import matplotlib.pyplot as plt
x=numpy.linspace(1,10,1000)
# Funcion de densidad
plt.subplot(131); plt.plot(pareto.pdf(x,5))
# Funcion de distribucion
plt.subplot(132); plt.plot(pareto.cdf(x,5))
# Generador aleatorio
plt.subplot(133);
plt.plot(pareto.rvs(5,size=1000))
plt.show()
```

Ejercicio 2

- Dibuja las funciones de densidad, de distribución y 1000 muestras de una distribución de probabilidad uniforme en [2,3]. Debes obtener el resultado que se muestra en esta transparencia.



scipy-3

- 25 sujetos emplearon un actuador que puede moverse hacia la izquierda o hacia la derecha. La siguiente tabla muestra los tiempos en segundos que cada sujeto empleó en mover el actuador una cierta distancia:

Subject	1	2	3	4	5	6	7	8	9	10
Right thread	113	105	130	101	138	118	87	116	75	96
Left thread	137	105	133	108	115	170	103	145	78	107
Subject	11	12	13	14	15	16	17	18	19	20
Right thread	122	103	116	107	118	103	111	104	111	89
Left thread	84	148	147	87	166	146	123	135	112	93
Subject	21	22	23	24	25					
Right thread	78	100	89	85	88					
Left thread	76	116	78	101	123					

- Se desea saber si las diferencias entre los tiempos empleados en los movimientos a la izquierda y a la derecha son significativas o si se deben al azar.

scipy-3

- Resultado del test e histograma

```
import numpy
import matplotlib.pyplot as plt
from scipy.stats import ttest_1samp
data = numpy.array([[113,105,130,101,138,118,87,116,75,96,
    122,103,116,107,118,103,111,104,111,89,78,100,89,85,88],
    [137,105,133,108,115,170,103,145,78,107, \
    84,148,147,87,166,146,123,135,112,93,76,116,78,101,123]])
dataDiff = data[1,:]-data[0,:]
print(dataDiff.mean(), dataDiff.std())
# Histograma
plt.hist(dataDiff)
plt.show()
# Test t
t_stat,p_value=ttest_1samp(dataDiff,0)
print("El p-valor es: %02f" % p_value)
if p_value<0.05:
    print("Los tiempos son significativamente diferentes")
else:
    print("No hay evidencia para rechazar: los actuadores son indistintos")
```

Ejercicio 3

- Usa un test de Wilcoxon para realizar la misma comprobación. ¿Cuál es el p-valor? ¿La conclusión es la misma, o es diferente?

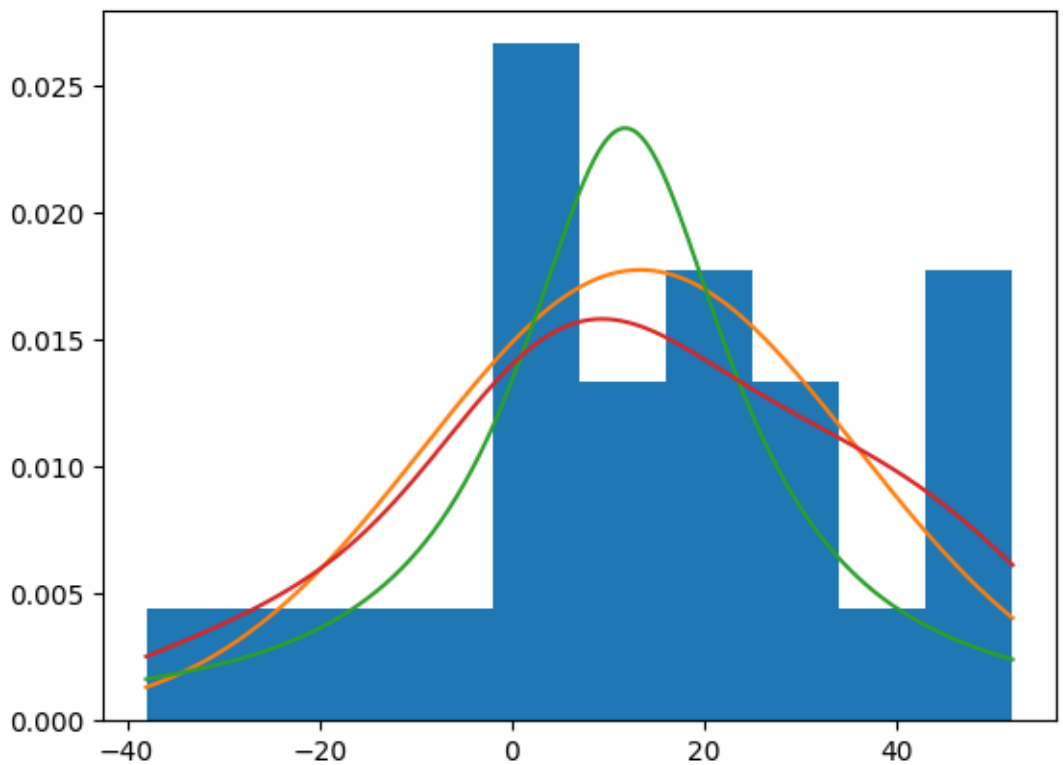
scipy-4

- Ajuste de una gaussiana a la diferencia entre los tiempos para cada individuo. Estimación kernel de la densidad.

```
import numpy
import matplotlib.pyplot as plt
import scipy.stats as ss
data = numpy.array([[113,105,130,101,138,118,87,116,75,96,
    122,103,116,107,118,103,111,104,111,89,78,100,89,85,88],
    [137,105,133,108,115,170,103,145,78,107, \
    84,148,147,87,166,146,123,135,112,93,76,116,78,101,123]])
dataDiff = data[1,:]-data[0,:]
# Ajuste de una normal a los datos
mean,std=ss.norm.fit(dataDiff)
plt.hist(dataDiff,density=1)
x=numpy.linspace(dataDiff.min(),dataDiff.max(),1000)
pdf=ss.norm.pdf(x,mean,std)
kde=ss.gaussian_kde(dataDiff)
plt.plot(x,pdf)
plt.plot(x,kde(x))
plt.show()
```

Ejercicio 4

- Ajusta también una distribución de Cauchy a los mismos datos. Debes obtener el resultado mostrado en la transparencia (curva verde)



scipy-5

- Distancias entre vectores: Euclídea,disimilaridad de Bray-Curtis, Canberra, Chebyshev, Manhattan, de correlación,del coseno, Sorensen-Dice, Hamming, Jaccard-Needham, Kulsinski, Mahalanobis, etc.

```
import numpy
import matplotlib.pyplot as plt
from scipy.spatial.distance import minkowski
Square=numpy.meshgrid(numpy.linspace(-1.1,1.1,512),
numpy.linspace(-1.1,1.1,512),indexing='ij')
X=Square[0]; Y=Square[1]
f=lambda x,y,p: minkowski([x,y],[0.0,0.0],p)<=1.0
Ball=lambda p:numpy.vectorize(f)(X,Y,p)
plt.imshow(Ball(3)); plt.axis('off'); plt.show()
```

Ejercicio 5

- Dibuja la bola de tamaño 1 para las distancias de Minkowski de órdenes 1,2 y 4

- Dibuja la bola de tamaño 1 para las distancias euclídea, Chebyshev y Manhattan

Parte 2: sklearn

- scikit-klearn incluye la mayoría de las herramientas de aprendizaje de máquina
- El flujo de trabajo en sklearn es (ver tutorial en clase de teoría):
    1. Cargar dataset
    2. Preprocesar dataset
    3. Aprender el modelo (método “fit()”)
    4. Evaluar el modelo (método “predict()”)

sklearn

- Caso de uso: problema “Boston” (UCI)

- Valor de las casas en los suburbios de Boston

1. CRIM: per capita crime rate by town
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town
4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. NOX: nitric oxides concentration (parts per 10 million)
6. RM: average number of rooms per dwelling
7. AGE: proportion of owner-occupied units built prior to 1940
8. DIS: weighted distances to five Boston employment centres
9. RAD: index of accessibility to radial highways
10. TAX: full-value property-tax rate per $10,000
11. PTRATIO: pupil-teacher ratio by town
12. B: 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT: % lower status of the population
14. MEDV: Median value of owner-occupied homes in $1000's

sklearn-1

```
boston_dataset = datasets.load_boston()
print(boston_dataset.feature_names)
X_full = boston_dataset.data
Y = boston_dataset.target
print(X_full.shape)
print(Y.shape)
# Se elige la variable mas dependiente de la salida
selector = SelectKBest(f_regression, k=1)
selector.fit(X_full, Y)
X = X_full[:, selector.get_support()]
print(X.shape)
plt.scatter(X, Y, color='black')
plt.show()
regressor = LinearRegression(normalize=True)
regressor.fit(X, Y)
plt.scatter(X, Y, color='black')
plt.plot(X, regressor.predict(X), color='blue',
linewidth=3)
plt.show()
```

Ejercicio 6

- sklearn-1-mod.py muestra un gráfico de la predicción del modelo lineal, la regresión SVM y el Random Forest cuando se utilizan todas las características.

- Como no es posible dibujar un gráfico scatter cuando el eje X tiene más de una dimensión, se ordenan las instancias de acuerdo con el valor de la variable independiente

- Modifica sklearn-1-mod.py para que se muestre cualquier otro modelo de regresión (polinomio, red neuronal, u otra cualquiera) y superpon las gráficas de LinearRegression, SVR y RandomForestRegressor para comparar gráficamente los ajustes

- Prueba con diferentes tipos de kernel y valores de parámetro C en SVR()

- Prueba con valores de k diferentes de 1 y de 13 en SelectKBest y elige el número de características que crees más adecuado

sklearn-2

- Para evaluar el modelo por validación cruzada se utiliza cross_val_score o cross_val_predict seguido de una métrica

```
regressor = LinearRegression(normalize=True)
regressor.fit(X, Y)
score = cross_val_score(regressor, X, Y).mean()
predicted = cross_val_predict(regressor, X, Y)
mse = mean_squared_error(Y,predicted)
print("LIN MSE=",mse)
print("LIN score=",score)
```

Ejercicio 7

- Cuál es el mejor modelo, LIN, SVR o RandomForest?

- Y si se emplean todas las variables en lugar de la más dependiente, el resultado es mejor o peor?

Parte 3: pandas

- Entrada de datos
- Visión general del dataset
- Manejo básico de valores perdidos
- Combinación de dataframes
- Selección de datos en un dataframe

Parte 3: pandas / entrada de datos

Archivo pandas-1.csv

- Lectura de datos desde csv
- Lectura de datos desde excel
- Creación de un dataframe desde un diccionario

Parte 3: pandas / visión general

Archivo pandas-2.csv

- Descripción estadística de datos numéricos (media, percentiles, etc)

Parte 3: pandas / valores perdidos

Archivo pandas-3.csv

- Datasets con datos perdidos
- Eliminar valores perdidos
- Reemplazar (imputar) valores perdidos

Parte 3: pandas / combinación de dataframes

Archivo pandas-4.csv

- Concatenar filas o columnas de datasets

Parte 3: pandas / selección de datos

Archivo pandas-5.csv

- Selección por índice
- Selección por etiqueta

Ejercicio 8

- Carga el dataset Churn_Modelling_NANs.xls y haz los siguientes pasos:

    1. Elimina las filas con valores perdidos
    2. Selecciona un dataframe 'X' con las columnas 'CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActiveMember'
    3. Selecciona un dataframe 'Y' con la columna 'EstimatedSalary'
    4. Haz tres modelos diferentes (por ejemplo, regresión lineal, SVR y Random Forest) de Y frente a X y compara el error cuadrático medio de los tres con validación cruzada (10 fold).
    5. (Opcional) Selecciona un dataframe 'XC' con las columnas 'CreditScore', 'Age',' Tenure', 'Balance', 'NumOfProducts',' HasCrCard', 'IsActiveMember', 'EstimatedSalary' y un dataframe 'C' con la columna 'Exited'. Haz tres clasificadores diferentes de C frente a XC y compara sus porcentajes de aciertos con validación cruzada (10 fold)
    6. (Opcional) En lugar de eliminar las filas con valores perdidos, prueba diferentes métodos de imputación y repite los puntos (4) y (6). Discute las diferencias en los resultados.
    7. (Opcional) Cuál crees que es la variable que más influye en el modelo del salario? Y en el modelo de la tasa de abandono?