Profesor: Dr. Oldemar Rodríguez Rojas
PROMiDAT Iberoamericano
Ciencia de Datos con Python

- Las tareas tienen fecha de entrega una semana después a la clase y deben ser entregadas antes del inicio de la clase siguiente.
- Cada día de atraso en implicará una pérdida de 10 puntos.
- Las tareas son estrictamente de carácter individual, tareas iguales se les asignará cero puntos.
- En nombre del archivo debe tener el siguiente formato: `Tarea1_nombre_apellido.pdf`. Por ejemplo, si el nombre del estudiante es Luis Pérez: `Tarea1_luis_perez.pdf`. Para la tarea número 2 sería: `Tarea2_luis_perez.pdf`, y así sucesivamente.
- Todas las preguntas tienen el mismo valor.
- Esta tarea tiene un valor de un 12.5 % respecto a la nota total del curso.

Tarea Número 7

- **Ejercicio 1:** [20 puntos] Supongamos que tenemos un modelo predictivo para detectar Fraude en Tarjetas de Crédito, la variable a predecir es `Fraude` con dos posibles valores `Sí` (para el caso en que sí fue fraude) y `No` (para el caso en que no fue fraude). Supongamos la matriz de confusión es:

	`No`	`Sí`
`No`	83254	12
`Sí`	889	3

  - Calcule a mano o en Excel la Precisión Global, la Precisión Positiva, la Precisión Negativa, los Falsos Positivos, los Falsos Negativos, la Asertividad Positiva y la Asertividad Negativa.
  - ¿Es bueno o malo el modelo predictivo? Justifique su respuesta.

- **Pregunta 2:** [40 puntos] Esta pregunta utiliza los datos sobre la conocida historia y tragedia del Titanic, usando los datos (`titanic.csv`) de los pasajeros se trata de predecir la supervivencia o no de un pasajero.

  La tabla contiene 12 variables y 1309 observaciones, las variables son:

  - `PassegerId`: El código de identificación del pasajero (valor único).
  - `Survived`: Variable a predecir, 1 (el pasajero sobrevivió) 0 (el pasajero no sobrevivió).
  - `Pclass`: En que clase viajaba el pasajero (1 = primera, 2 = segunda , 3 = tercera).
  - `Name`: Nombre del pasajero (valor único).
  - `Sex`: Sexo del pasajero.
  - `Age`: Edad del pasajero.
  - `SibSp`: Cantidad de hermanos o cónyuges a bordo del Titanic.

- `Parch`: Cantidad de padres o hijos a bordo del Titanic.
- `Ticket`: Número de tiquete (valor único).
- `Fare`: Tarifa del pasajero.
- `Cabin`: Número de cabina (valor único).
- `Embarked`: Puerto donde embarco el pasajero (C = Cherbourg, Q = Queenstown, S = Southampton).

1. Cargue la tabla de datos `titanic.csv` en **Python**. Asegúrese re-codificar las variables cualitativas y de ignorar variables que no se deben usar.
2. Genere al azar una tabla de testing con el 25 % de los datos y con el resto de los datos genere una tabla de aprendizaje.
3. Usando todos los métodos predictivos vistos en clase genere modelos predictivos para la tabla de aprendizaje. Grafique el árbol de decisión.
4. Con la tabla de testing, para todos los modelos generados en el punto anterior, calcule la matriz de confusión, la precisión, la precisión positiva, la precisión negativa, los falsos positivos, los falsos negativos, la acertividad positiva y la acertividad negativa. Luego construya un cuadro comparativo ¿Cuál método es mejor?
5. Repita los ejercicios anteriores pero para Bosques Aleatorios y Potenciación use solamente las 3 variables que en los respectivos modelos tuvieron mayor importancia.

- **Pregunta 3:** [40 puntos] **Reconocimiento Facial.** En este ejercicio usaremos el archivo de datos `8carasFamosas.csv` el cual contiene 1176 fotos (individuos) de 8 personas diferentes y 2915 variables, donde las primeras 2914 variables representan un pixel de la foto y la variable restante es la variable a predecir, es decir, el nombre de la persona. Es decir, se tienen 2914 variables que son los pixeles de la foto, cada fila representa realmente una matriz de 47×62, solamente que fue "estirada" en $47 \times 62 = 2914$ variables. A continuación se muestra parcialmente como se ve realmente una fila:

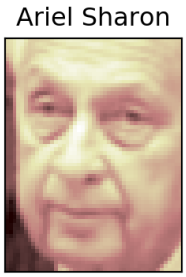
Filas	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
1	155.666672	151.666672	150.33333	151.333328	157.000000	156.00000	143.33333	117.33334	93.666664	83.000000
2	152.333328	149.000000	152.00000	159.333328	157.666672	133.33333	105.33334	87.00000	78.333336	73.666664
3	150.666672	154.333328	159.66667	151.000000	123.333336	92.33334	79.00000	76.33334	74.666664	70.666664
4	154.333328	160.000000	146.66667	110.000000	75.000000	74.33334	77.00000	71.66666	72.666664	70.666664
5	155.333328	141.000000	100.33334	67.666664	62.333332	69.66666	74.33334	73.33334	73.000000	74.666664
6	130.000000	91.333336	61.66667	58.666668	65.333336	68.66666	74.33334	76.00000	75.333336	79.333336
7	83.333336	60.000000	58.00000	65.333336	66.666664	69.66666	77.00000	81.00000	82.333336	87.000000
8	60.000000	62.333332	65.66666	68.000000	72.000000	73.66666	82.00000	89.66666	91.000000	93.000000
9	61.666668	67.000000	70.00000	73.000000	75.333336	81.33334	91.33334	96.33334	95.666664	94.666664
10	67.333336	68.000000	73.33334	77.000000	81.333336	88.33334	96.00000	96.66666	93.333336	90.333336
11	68.000000	67.333336	72.66666	78.333336	83.000000	88.33334	90.66666	88.66666	87.666664	89.333336
12	65.333336	68.333336	71.66666	76.666664	81.000000	82.66666	81.66666	81.00000	85.333336	91.000000
13	66.666664	72.333336	73.66666	76.000000	79.333336	78.66666	78.33334	81.33334	87.333336	92.333336
14	67.000000	70.333336	71.00000	72.000000	74.000000	75.00000	76.00000	79.00000	84.333336	89.666664
15	64.666664	66.666664	68.66666	69.666664	70.666664	74.33334	78.00000	78.00000	78.666664	80.666664

Con la siguiente función podemos graficar una foto (fila de la tabla de datos):

```
def plot_image(valor_cara, titulo = None, filas = 62, cols = 42):
    image = np.array(list(reversed(valor_cara)))
    image = pd.to_numeric(image, errors = 'coerce')
    image = image.reshape(62, 47)
    plt.imshow(image, cmap = "pink")
    ejes = plt.gca()
    ejes.axes.get_xaxis().set_visible(False)
    ejes.axes.get_yaxis().set_visible(False)
    if titulo is not None:
        plt.title(titulo)
```

Así, se puede graficar una foto como sigue:

```
plot_image(datos_caras.iloc[0, range(2914)], datos_caras.iloc[0, 2914])
```



Con el siguiente código podemos imprimir varias fotos:

```
import random
for i in range(1, 9):
    plt.subplot(2, 4, i)
    cara = random.randint(0, datos_caras.shape[0])
    plot_image(datos_caras.iloc[cara, range(2914)], datos_caras.iloc[cara, 2914])
```

Con estos datos realice lo siguiente:

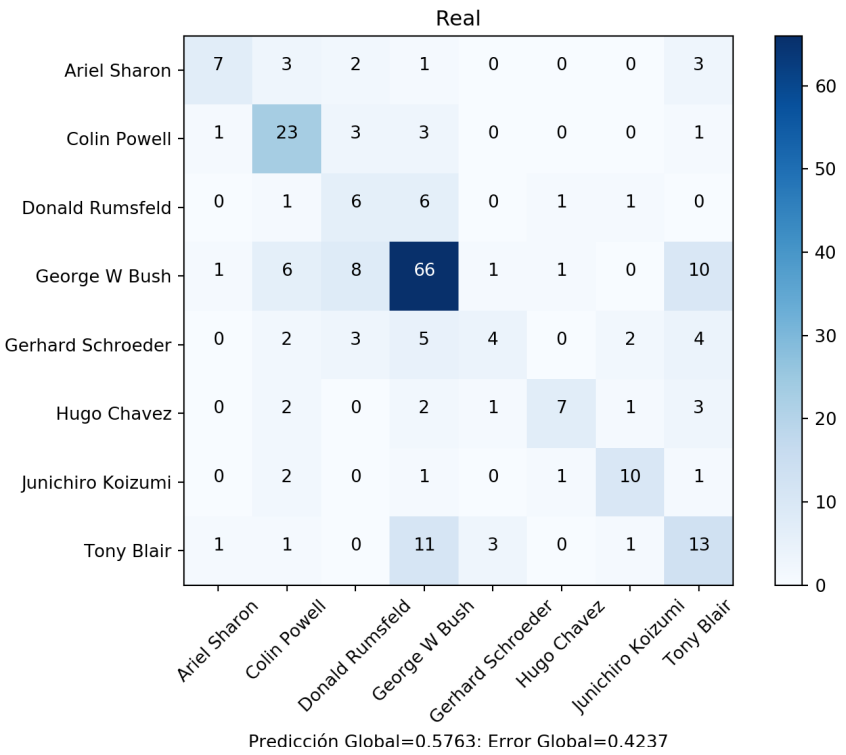
1. Realice una clusterización jerárquica usando la distancia `euclidean` y el método (índice de agregación) `ward`. Grafique el dendograma con 8 clústeres y luego grafique la foto que representa el centro de gravedad de cada uno de estos 8 clústeres. ¿Se ven nítidas las fotos? ¿Se pueden identificar el personaje? Explique por qué sí o por qué no.
2. Genere en **Python** la tabla de testing con el 20 % de los datos y con el resto de los datos genere una tabla de aprendizaje.
3. Usando todos los métodos predictivos vistos en clase genere modelos predictivos para la tabla de aprendizaje. Grafique el árbol de decisión.
4. Usando la función `indices_general(...)` vista en clase para la tabla de testing calcule las matrices de confusión y plotee dichas matrices usando la función `plot_confusion_matrix(...)` cuyo código viene más abajo y pueden bajar el archivo `plot_confusion_matrix.py` del aula virtual.
5. ¿Es buena la predicción? ¿Para cuáles personajes la predicción es buena y para cuáles es mala?
6. Repita los ejercicios anteriores `KNeighborsClassifier`, `DecisionTreeClassifier`, `RandomForestClassifier`, `AdaBoostClassifier` y `GradientBoostingClassifier`, pero esta vez cambie algunos de los parámetros de manera que los resultados mejoren.

La siguiente función grafica la matriz de confusión con los valores de la precisión de cada categoría y la precisión global. Pueden bajar esta función en el archivo `plot_confusion_matrix.py` del aula virtual.

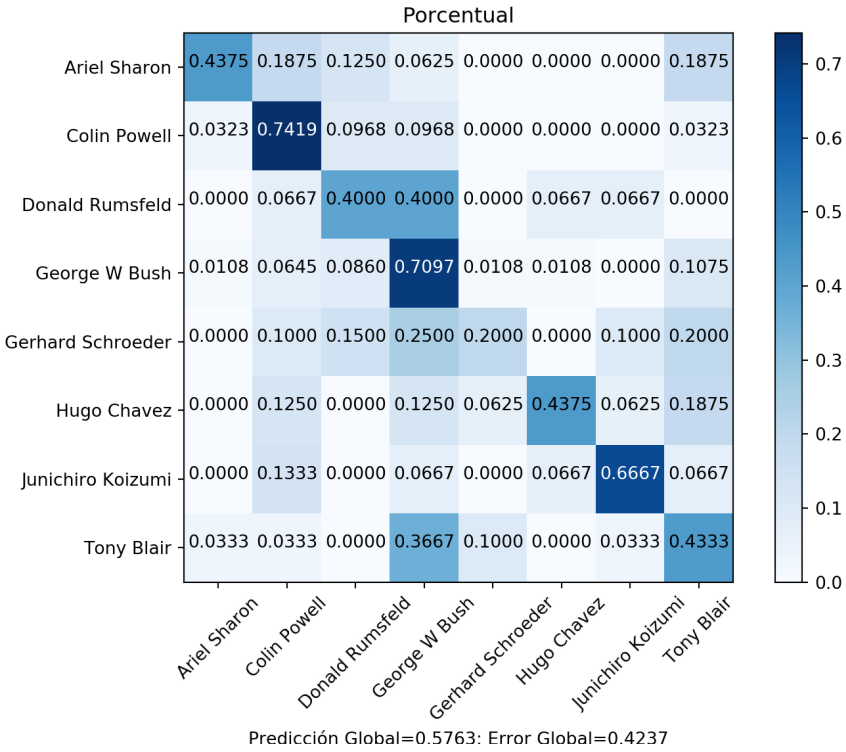
```
def plot_confusion_matrix(cm, target_names, title = 'Matriz de Confusión',
                                                cmap = None, normalize = True):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools

    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy
    if cmap is None:
        cmap = plt.get_cmap('Blues')

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    plt.figure(figsize = (8, 6))
    plt.imshow(cm, interpolation = 'nearest', cmap = cmap)
    plt.title(title)
    plt.colorbar()

    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation = 45)
        plt.yticks(tick_marks, target_names)

    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.4f}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.xlabel('Precisión Global={:0.4f};
                     Error Global={:0.4f}'.format(accuracy, misclass))
```

Ejemplos de uso:

```
MC = confusion_matrix(y_test, modelo_knn.predict(X_test))
plot_confusion_matrix(MC, unique_labels(datos_caras['Nombres']), title = "Real",
                                        normalize = False)
```




```
plot_confusion_matrix(MC, unique_labels(datos_caras['Nombres']),
                                        title = "Porcentual")
```



Entregables:

1. Genere desde `Jupyter Notebook` un documento autoreproducible con la solución de la tarea y súbalo en el Aula Virtual.