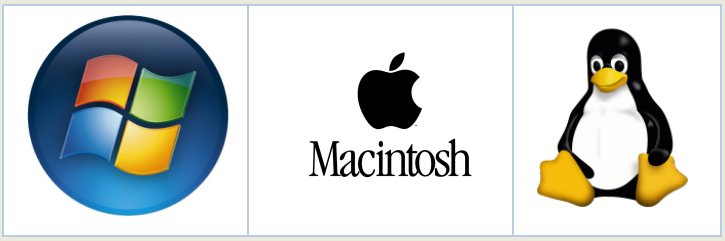
ML_CLASIFICACION_REGLOGISTICA_01	Modelo para predecir el tipo de usuario de SO	ML

Se dispone de una base de datos con información de usuarios que utilizan internet desde diferentes sistemas operativos (SO) como Windows, Macintosch y Linux. La información consiste en la duración (en segundos) que los usuarios están conectados, el número de páginas que visitan, las acciones (click, scroll,…) durante la sesión y la suma de la valoración de las acciones (cada acción lleva asociada un valor).



El objetivo de la práctica es construir un modelo de aprendizaje maquina (machine learning), basado en el algoritmo de ***Regresión Logística*** que aprenda a clasificar el sistema operativo que utilizan los usuarios conocida la duración, número de páginas, acciones y valoración.

Predecir el sistema operativo que utiliza un usuario si tiene las siguientes características:

Duración (en segundos): 5

Páginas visitadas: 2

Acciones: 3

Valoración: 5

SOLUCIÓN

Importar las librerías necesarias para realizar la práctica.

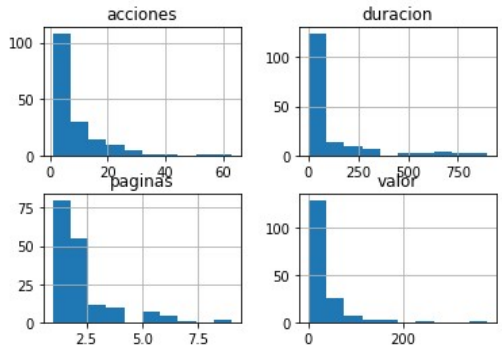
```
# Librerías
import pandas as pd
import numpy as np
from sklearn import linear_model
from sklearn import model_selection
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sb
```

Cargar la base de datos (en formato csv).

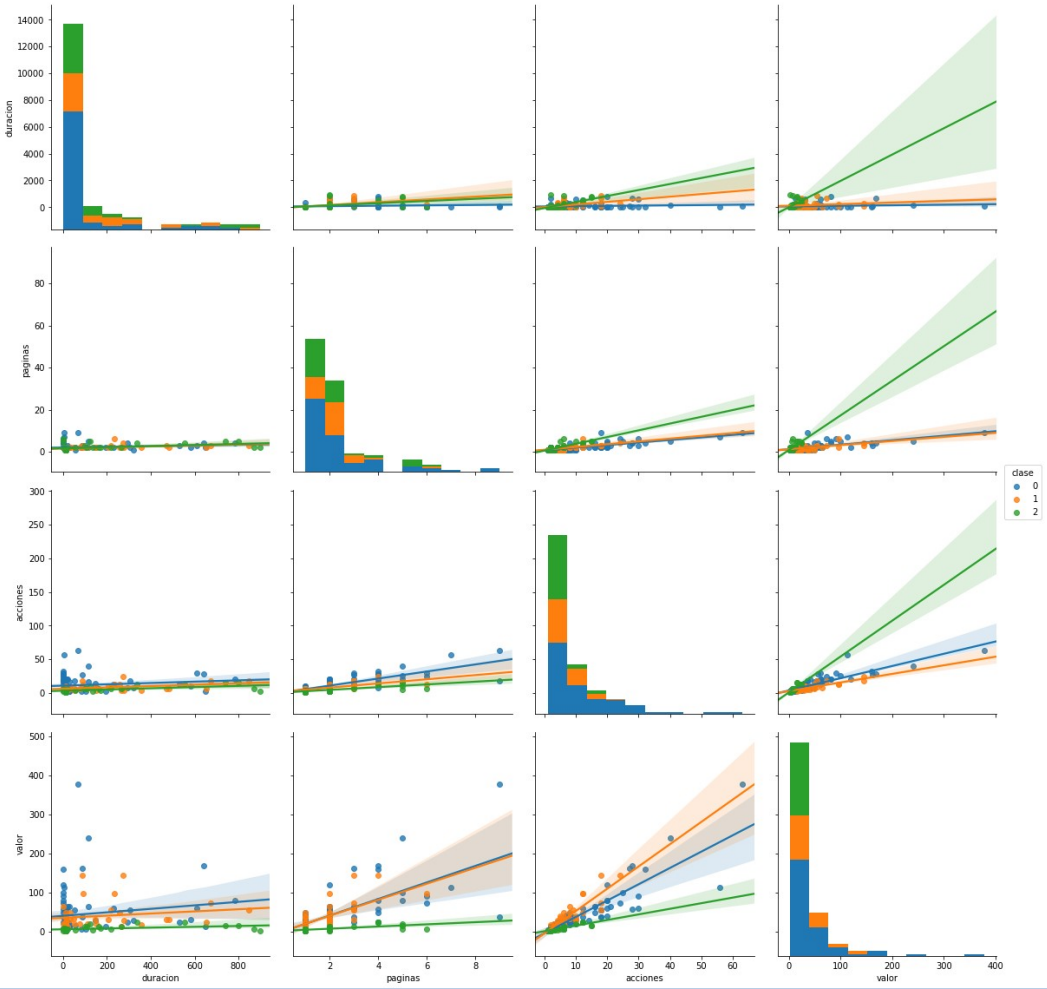
```
# Cargar la información del fichero CSV
dataframe = pd.read_csv(r"usuarios_win_mac_lin.csv")
```

Visualizar la información de la base de datos:

```
# Visualizar las 5 primeras filas del fichero
dataframe.head()

# Clases de usuarios:  0 -> Windows, 1-> Macintosh, 2-> Linux

# Consultar información de la base de datos
dataframe.describe()

# Analizar cuantos ejemplos existen de cada clase
print(dataframe.groupby('clase').size())

clase
0    86
1    40
2    44
dtype: int64

# Visualizar datos
dataframe.drop(['clase'],1).hist()
plt.show()
```




```
# Visualizar por pares de atributos con la librería seaborn
sb.pairplot(dataframe.dropna(), hue='clase',size=4,vars=["duracion",
"paginas","acciones","valor"],kind='reg')
```



Crear y dividir las variables X e y:

```
# Definir X e y
X = np.array(dataframe.drop(['clase'],1))
y = np.array(dataframe['clase'])
X.shape


# Dividir para entrenamiento y test (80 % para entrenamiento y 20 % para
validación)
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,
test_size=0.2, random_state=7)
```

Crear el clasificador basado en Regresión Logística.

```
# Crear y ajustar el modelo de regresión logística
model = linear_model.LogisticRegression()
```

Ajustar el modelo.

```
# Ajustar el modelo
model.fit(X_train,y_train)
```

Evaluar con la matriz de confusión:

```
# Evaluación con la matriz de confusión
print("Matriz de Confusión")
cm=confusion_matrix(y_test, y_pred)
print(cm)
print()

print("Verdaderos positivos (VP) (Windows -> Windows):",cm[0][0])
print("Falsos positivos (FP) (Windows -> Macintosh):",cm[1][0])
print("Falsos positivos (FP) (Windows -> Linux):",cm[2][0])

print("Verdaderos positivos (VP) (Macintosh -> Macintosh):",cm[1][1])
print("Falsos positivos (FP) (Macintosh -> Windows):",cm[0][1])
print("Falsos positivos (FP) (Macintosh -> Linux):",cm[2][1])

print("Verdaderos positivos (VP) (Linux -> Linux):",cm[2][2])
print("Falsos positivos (FP) (Linux -> Windows):",cm[0][2])
print("Falsos positivos (FP) (Linux -> Macintosh):",cm[1][2])

Matriz de Confusión
[[14  2  2]
 [ 3  3  0]
 [ 0  0 10]]
```

```
Verdaderos positivos (VP) (Windows -> Windows): 14
Falsos positivos (FP) (Windows -> Macintosh): 3
Falsos positivos (FP) (Windows -> Linux): 0
Verdaderos positivos (VP) (Macintosh -> Macintosh): 3
Falsos positivos (FP) (Macintosh -> Windows): 2
Falsos positivos (FP) (Macintosh -> Linux): 0
Verdaderos positivos (VP) (Linux -> Linux): 10
Falsos positivos (FP) (Linux -> Windows): 2
Falsos positivos (FP) (Linux -> Macintosh): 0
```

Generar el informe de clasificación:

```
# Informe de clasificación
from sklearn.metrics import classification_report

print()
print(" Informe de clasificación (sobre datos de test")
print(classification_report(y_test,y_pred))
```

```
Informe de clasificación (sobre datos de test
             precision    recall  f1-score   support

          0       0.82      0.78      0.80        18
          1       0.60      0.50      0.55         6
          2       0.83      1.00      0.91        10

avg / total       0.79      0.79      0.79        34
```

Predecir para un nuevo usuario:

```
# PREDECIR PARA UN CASO DETERMINADO
# Suponer un usuario con valores: Tiempo Duración: 5, Paginas visitadas: 2,
Acciones al navegar: 3, Valoración: 5
X_nuevo = pd.DataFrame({'duracion': [5], 'paginas': [2], 'acciones': [3],
'valor': [5]})
y_pred_nuevo=model.predict(X_nuevo)
print("Nuevo usuario: ",y_pred_nuevo)
```

```
Nuevo usuario:  [2]
```