Inteligencia Artificial
Regresión Logística

Nombre: Bárbara Rossana Pérez Silva
Grupo: 031 **2015576**

1 Introducción

La Regresión Logística es un modelo matemático el cual crea una relación entre un par de datos con el fin de predecir uno de estos factores. A partir de un conjunto de datos deentrada(características), nuestrasalida será discreta (y no continua como es usual en la Regresión Lineal y Lineal Múltiple). La predicción que obtenemos es de un número finito de estados, ya sea binario, como un SI/NO, o una selección de etiquetas o clases, múltiple.

2 Metodología

Para esta nueva práctica necesitamos descargar el archivo .csv que se nos otorga, el contexto que tenemos es el siguiente:
Datos de entrada para clasificar si el usuario que visita un sitio web usa como sistema operativo Windows, Macintosh o Linux. Nuestra información de entrada son 4 características:

- *Duración de la visita en Segundos*
- *Cantidad de Páginas Vistas durante la Sesión*
- *Cantidad de Acciones del usuario (click, scroll, uso de checkbox, sliders, etc.)*
- *Suma del Valor de las acciones (cada acción lleva asociada una valoración de importancia)*

Como la salida es discreta, asignaremos los siguientes valores a las etiquetas:
0- Windows
1- Macintosh
2-Linux

Ahora para iniciar nuestro código necesitamos importar ciertas librerías, pero en esta ocasión no hacemos uso de la línea de comando *pip install ...* ya que estaremos utilizando librerías ya instaladas anteriormente.

```
1  import pandas as pd
2  import numpy as np
3  from sklearn import linear_model
4  from sklearn import model_selection
5  from sklearn.metrics import classification_report
6  from sklearn.metrics import confusion_matrix
7  from sklearn.metrics import accuracy_score
8  import matplotlib.pyplot as plt
9  import seaborn as sb
```

Lo siguiente es leer el archivo .csv y cargarlo como un dataset de Pandas; con la línea de código .head() podremos visualizar esos primeros registros.

```
11  dataframe = pd.read_csv(r"usuarios_win_mac_lin.csv")
12  print(dataframe.head())
```

Con la función .describe(), como ya lo habíamos visto en Regresión Lineal, nos mosstrará estadísticas de nuestros nuevos datos.

```
14  print(dataframe.describe())
```

Ahora con la función *groupby* podemos visualizar cuantos usuarios tenemos registrados de cada sistema operativo.

```
16  print(dataframe.groupby('clase').size())
```

Podemos visualizar los datos que tenemos con ayuda de histogramas, esto nos puede llegar a beneficiar a la hora de comprender mejor como se distribuyen los datos conforme a su mínimo y máximo.

```
18  dataframe.drop(['clase'], axis= 1).hist()
19  plt.show()
```

También podemos interrelacionar las entradas de a pares, para ver como se concentran linealmente las salidas de usuarios por colores: Sistema Operativo Windows en azul, Macintosh en verde y Linux en rojo.

```
21  sb.pairplot(dataframe.dropna(), hue = 'clase', height = 4,vars=[
22      "duracion", "paginas","acciones","valor"], kind = 'reg')
23  plt.show()
```

Cargaremos las columnas 4 columnas que tenemos como datos de entrada para nuestra variable X, mientras que asignaremos la columna "clase" para la variable Y; ejecutamos la función *.shape* con el fin de comprobar la dimensión de nuestra matriz de datos.

```
24  X = np.array(dataframe.drop(['clase'], axis = 1))
25  y = np.array(dataframe['clase'])
26  print(X.shape)
```

Entrenamos nuestro modelo con los datos que acabamos de recabar.

```
28  model = linear_model.LogisticRegression(max_iter=1000)
29  model.fit(X,y)
```

Una vez compilado nuestro modelo, le hacemos clasificar todo nuestro conjunto de entradas X, viendo que coincidan con las salidas reales de nuestro archivo csv.

```
31  predictions = model.predict(X)
32  print(predictions[0:5])
33  print(model.score(X,y))
```

Volvemos a validar nuestro modelo, ahora subdividiendo nuestros datos de entrada en un set de entrenamiento, este siendo el 80% de datos, y otro para validar el modelo, tomando unicamente el 20% de los datos.

```
35  validation_size = 0.20
36  seed = 7
37  X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(
38      X, y, test_size = validation_size, random_state = seed)
```

Nuevamente hacemos uso de la Regresión Logística con la diferencia de, que como vimos arriba, solo se usarán el 80% de los datos.

```
39  name = 'Logistic Regression'
40  kfold = model_selection.KFold(n_splits = 10, random_state = None)
41  cv_results = model_selection.cross_val_score(model, X_train, Y_train, cv = kfold, scoring = 'accuracy')
42  msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
43  print(msg)
```

Y mandamos a imprimir en pantalla nuestras predicciones, o clasificación, utilizando nuestro cross validation set, es decir del subconjunto que habíamos apartado.

```
45  predictions = model.predict(X_validation)
46  print(accuracy_score(Y_validation, predictions))
```

Ahora podemos visualizar con las siguientes líneas de código la matriz de confusión, la cual nos diría los resultados equivocados que obtuvimos de cada clase. Así como datos acerca de los resultados obtenidos con el reporte de clasificación.

```
48  print(confusion_matrix(Y_validation,predictions))
49  print(classification_report(Y_validation, predictions))
```

3 Resultados

	duracion	paginas	acciones	valor	clase
0	7.0	2	4	8	2
1	21.0	2	6	6	2
2	57.0	2	4	4	2
3	101.0	3	6	12	2
4	109.0	2	6	12	2

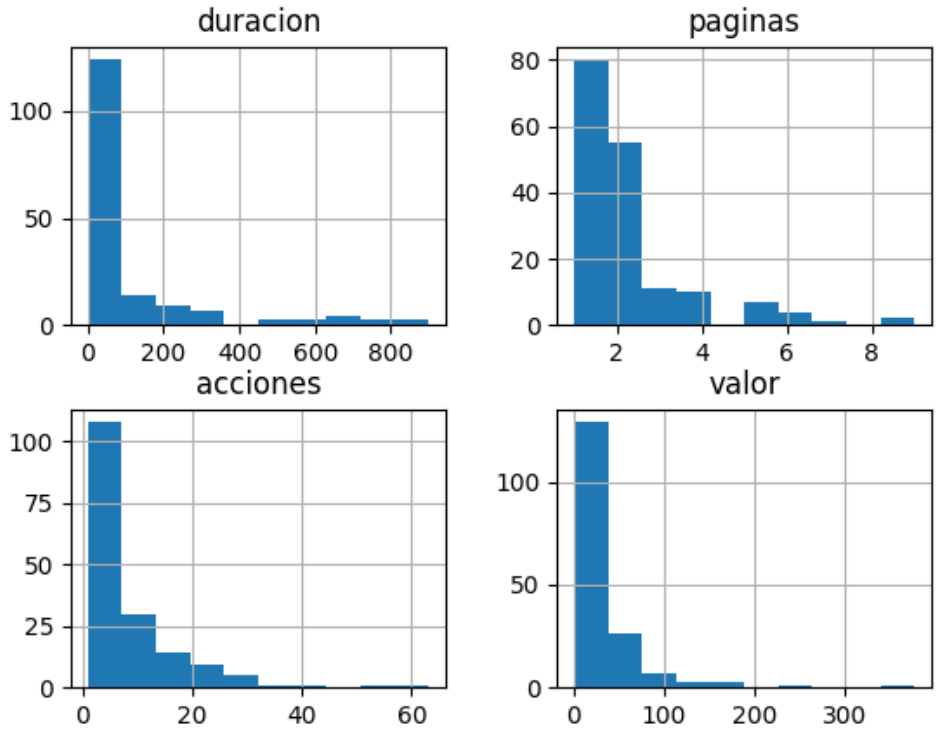
Podemos visualizar los primeros datos que tenemos en nuestro archivo .csv, estos acomodados en forma de tabla con las columnas de "duracion", "paginas", "acciones" y "valor".

	duracion	paginas	acciones	valor	clase
count	170.000000	170.000000	170.000000	170.000000	170.000000
mean	111.075729	2.041176	8.723529	32.676471	0.752941
std	202.453200	1.500911	9.136054	44.751993	0.841327
min	1.000000	1.000000	1.000000	1.000000	0.000000
25%	11.000000	1.000000	3.000000	8.000000	0.000000
50%	13.000000	2.000000	6.000000	20.000000	0.000000
75%	108.000000	2.000000	10.000000	36.000000	2.000000
max	898.000000	9.000000	63.000000	378.000000	2.000000

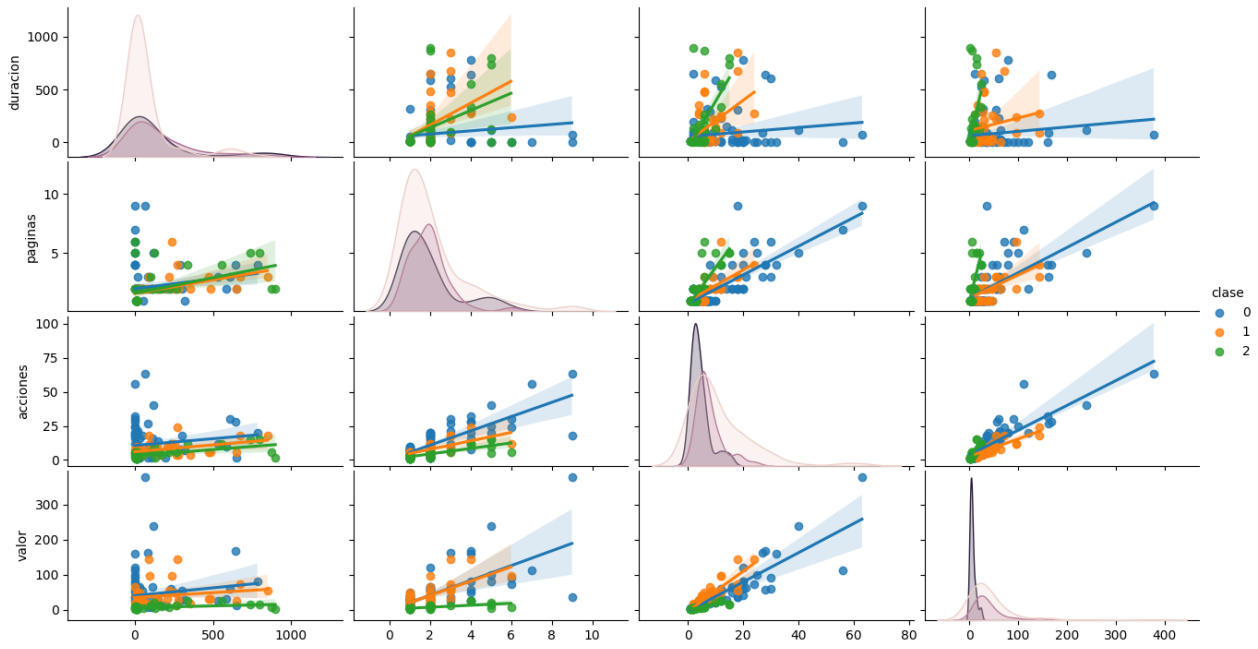
Vemos las estadísticas de los datos, cual es la cantidad mínima y máxima de cada columna, el promedio de cada una de ellas, entre otros datos adicionales.

```
clase
0    86
1    40
2    44
dtype: int64
```

Aqui se ve el recuento de usuarios de cada sistema operativo, teniendo 86 usuarios de Windows, 40 usuarios de Mac y 44 usuarios de Linux.



Podemos visualizar los datos de las cuatro características: "duracion", "paginas", "acciones" y "valor", estos inclinándose mayoritariamente hacía los valores mínimos de cada histograma.



Ahora tenemos la opción de visualizar los datos en diversos gráficos para ver como se concentran linealmente con respecto a las clases.

```
(170, 4)
```

Nuestra matriz de datos contiene 4 columnas, como bien lo establecimos, y una cantidad de 170 registros.

```
[2 2 2 2 2]
0.7764705882352941
```

Si analizamos nuestros datos, podemos verificar que nuestra predicción corresponde verídicamente, y podemos ver que en nuestro caso la precisión media de las predicciones es del 77%.

```
Logistic Regression: 0.728571 (0.094186)
```

En esta segunda corrida de la función de Regresión Logística, vemos que si usamos el 80% de los datos, tenemos un 72% de scoring.

```
0.8529411764705882
```

Podemos observar que el porcentaje de aciertos de nuestras predicciones fue del 85%, esto es un buen número, pero hay que tomar en cuenta que nuestro tamaño de datos era pequeño.

```
[[16  0  2]
 [ 3  3  0]
 [ 0  0 10]]
```

En nuestra matriz de confusión podemos ver que alguno de los errores que obtuvimos fue que predijo 3 usuarios que eran Mac como usuarios de Windows y que predijo a 2 usuarios Linux que realmente eran de Windows.

	precision	recall	f1-score	support
0	0.84	0.89	0.86	18
1	1.00	0.50	0.67	6
2	0.83	1.00	0.91	10
accuracy			0.85	34
macro avg	0.89	0.80	0.81	34
weighted avg	0.87	0.85	0.84	34

Podemos ver las métricas de "precision", "recall", "f1-score" y "support" de nuestro modelo de Regresión Logística.

Como un *plus*, vamos a inventar los datos de entrada de navegación de un usuario ficticio que tiene estos valores

- Tiempo Duración: 10
- Paginas visitadas: 3

- Acciones al navegar: 5
- Valoración: 9

Probamos el modelo prediciendo estos valores, obteniendo que el usuario se trata de uno con un sistema operativo Linux.

```
51  X_new = pd.DataFrame({'duracion': [10], 'paginas': [3], 'acciones': [5], 'valor': [9]})
52  prediction = model.predict(X_new.values)
53  print(prediction)
```

```
[2]
```

4 Conclusión

En esta práctica aprendí la importancia de la regresión logística, la cual unicamente había escuchado vagamente en el pasado, pero gracias a esta práctica conocí su significado y posible uso dentro del mundo de la programación; así como un ejemplo que leí en la página de Amazon *AWS* en donde mencionaban que la regresión logística en una empresa *"(se puede usar) para el análisis predictivo a fin de reducir los costos operativos, aumentar la eficiencia y escalar más rápido"*