Regresión Logística en Python

1 Introducción

La regresión logística es un método estadístico fundamental que modela la probabilidad de que una observación pertenezca a una categoría específica. A diferencia de la regresión lineal, la variable dependiente es categórica (binaria o multiclase). En la regresión logística estándar la variable dependiente solo puede tomar como valor 0 o 1, pero también existe la regresión logística multinomial que predice una probabilidad para todos los posibles valores de la variable dependiente. El modelo se basa en la función logística (sigmoide):

$$p(y = k) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{1}$$

donde:

- p es la probabilidad de que y tome un valor k
- y es la variable dependiente
- k es un valor binario: 0 o 1
- e es el número de Euler: 2.7182818284...
- x es la variable independiente
- β_0 es el intercepto
- β_1 es la pendiente

2 Metodología

Para realizar el ejercicio de regresión logística, tuve que seguir los siguientes pasos: Visualización General de los datos de Entrada, Creación y ajuste del modelo y Validación del Modelo.

```
#Importamos las librerias
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
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sb
%matplotlib inline

#Creamos un dataframe
dataframe = pd.read_csv(r"usuarios_win_mac_lin.csv")
```

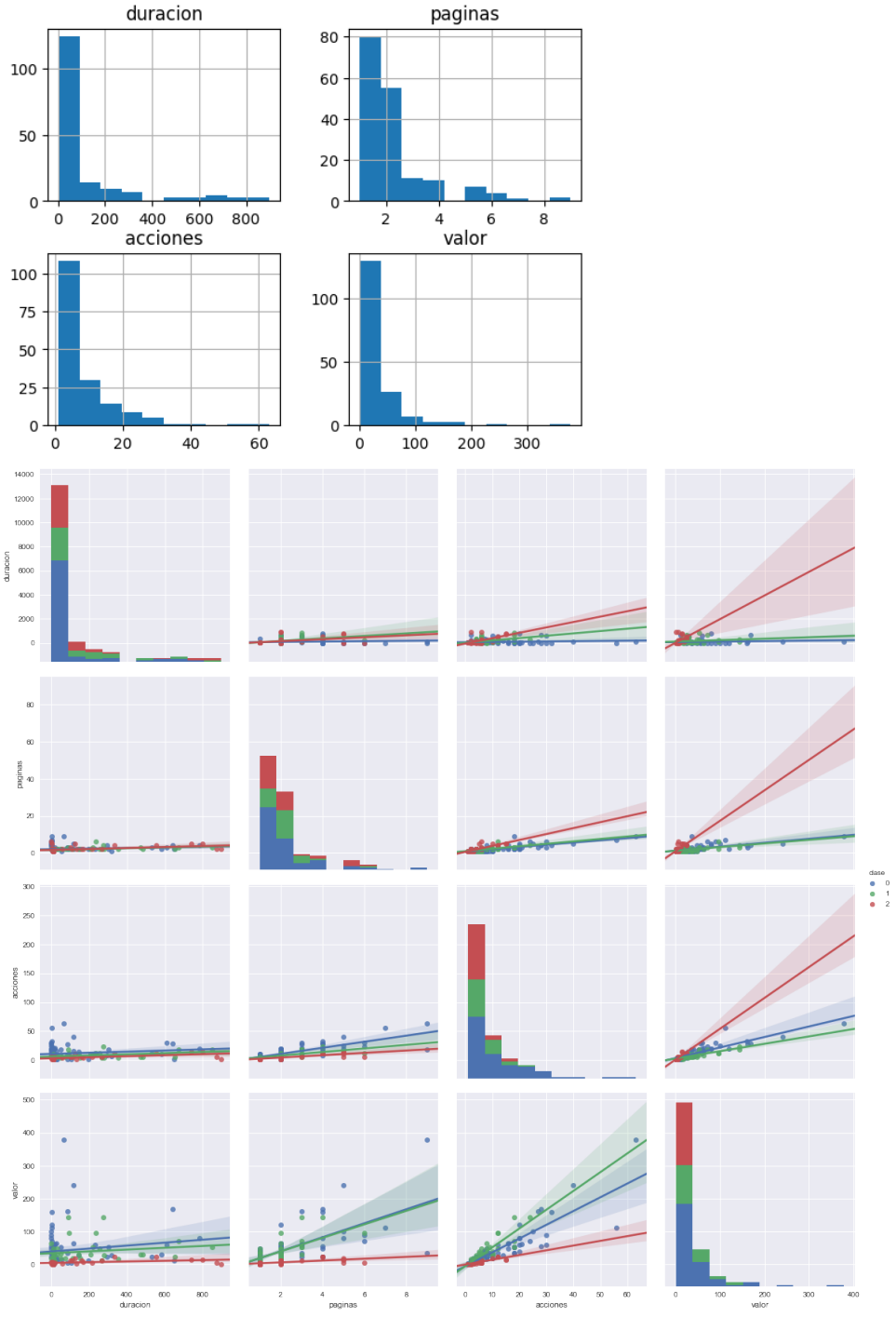
2.1 Visualización General de los datos de Entrada

En esta sección nos damos cuenta que la columna del *dataset* de clase tiene tres posibles valores: 0, 1 y 2; que corresponden con el sistema operativo del usuario. Esta variable va a ser nuestra variable dependiente que vamos a intentar predecir. Vamos a visualizar la distribución de los valores para cada columna y la relación entre las columnas.

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

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

Visualizamos la distribución de los valores de las columnas y la relación de cada columna entre sí:



2.2 Creación y ajuste del modelo

Para este modelo la variable dependiente es la columna de clase y las variables independientes son las otras cuatro columnas númericas del *dataset*. Establecemos los valores de X y Y y ajustamos el modelo de acuerdo a estos valores.

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

model = linear_model.LogisticRegression()
model.fit(X,y)
```

Después de esto, creamos las predicccciones de Y e imprimos los primeros 5 valores de este arreglo.:

```
predictions = model.predict(X)
print(predictions[0:5])
```

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

Finalmente, imprimimos el puntaje del modelo:

```
model.score(X,y)
```

```
0.7764705882352941
```

2.3 División de datos en conjuntos de entrenamiento

Asignamos los valores de nuestra columna *"Word Count"* como X, y los valores de la columna *"# Shares"* como nuestra Y.

```
# Asignamos nuestra variable de entrada X para entrenamiento y las etiquetas Y.
dataX = filtered_data[["Word count"]]
X_train = np.array(dataX)
y_train = filtered_data['# Shares'].values
```

2.4 Validación del modelo

En esta parte se intenta confirmar la precisión del modelo creando otro conjunto de datos que no son parte de los datos de entrenamiento, conocidos como datos de validación. Después utilizamos el método de validación cruzada para dividir los datos de entrenamiento en n partes y que cada una de estas se evalue con los datos de validación.

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

name='Logistic Regression'
kfold = model_selection.KFold(n_splits=10, shuffle=True, random_state=seed)
cv_results = model_selection.cross_val_score(model, X_train, Y_train,
    cv=kfold, scoring='accuracy')
msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
print(msg)
```

```
Logistic Regression: 0.720330 (0.151123)
```

Luego creamos las predicciones de este modelo validado y las imprimimos.

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

3 Resultados

Para checar los resultados, vamos a imprimir la matriz de confusión y el reporte de clasificación del conjunto de validación de Y y las predicciones.

```
print(confusion_matrix(Y_validation, predictions))
```

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

```
print(classification_report(Y_validation, predictions))
```

```
              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
```

4 Conclusión

Los modelos de regresión logística son herramientas estadísticas usadas para calcular la probabilidad que existe entre una variable dependiente binaria y un conjunto de variables independientes, en las que exista una relación logística entre estas. Hay dos tipos de regresión logística: estándar, que es cuando la variable dependiente solo puede tomar como valor 0 o 1; y multinomial, que es cuando puede tomar más valores.

En este caso, se creo un modelo logístico multinomial que intenta mostrar la relación entre la duración, páginas, acciones y valor de los usuarios y la clase de sistema operativo que usan. Al evaluar el modelo, imprimiendo su puntaje y luego validandolo usando técnicas más avanzadas para prevenir lo que se conoce como *overfitting*, podemos ver que el modelo tiene buena precisión para predecir el tipo de sistema operativo que usa el usuario.