# Regresión Logística

Jared Sandoval

March 23, 2025

## 1 Introducción

La regresión logística es un modelo de clasificación utilizado en aprendizaje automático para predecir la probabilidad de una clase en función de variables independientes. En este informe, aplicamos regresión logística para clasificar usuarios según su sistema operativo en base a datos de interacción con un sitio web.

## 2 Metodología

El proceso consistió en los siguientes pasos:

### 2.1 Importación de librerías y carga de datos

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

dataframe = pd.read_csv("usuarios_win_mac_lin.csv")
print(dataframe.head())
print(dataframe.describe())
print(dataframe.groupby('clase').size())
```

| | 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 |

Figure 1: datahead

| | 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 |

Figure 2: datadescribe

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

Figure 3: group by

## 2.2 Visualización de datos

```
dataframe.drop(['clase'], axis=1).hist()
plt.show()
sb.pairplot(dataframe.dropna(), hue='clase', size=4, vars
    =["duracion", "paginas", "acciones", "valor"], kind='
    reg')
plt.savefig("pairplot.png", dpi=300, bbox_inches='tight')
```

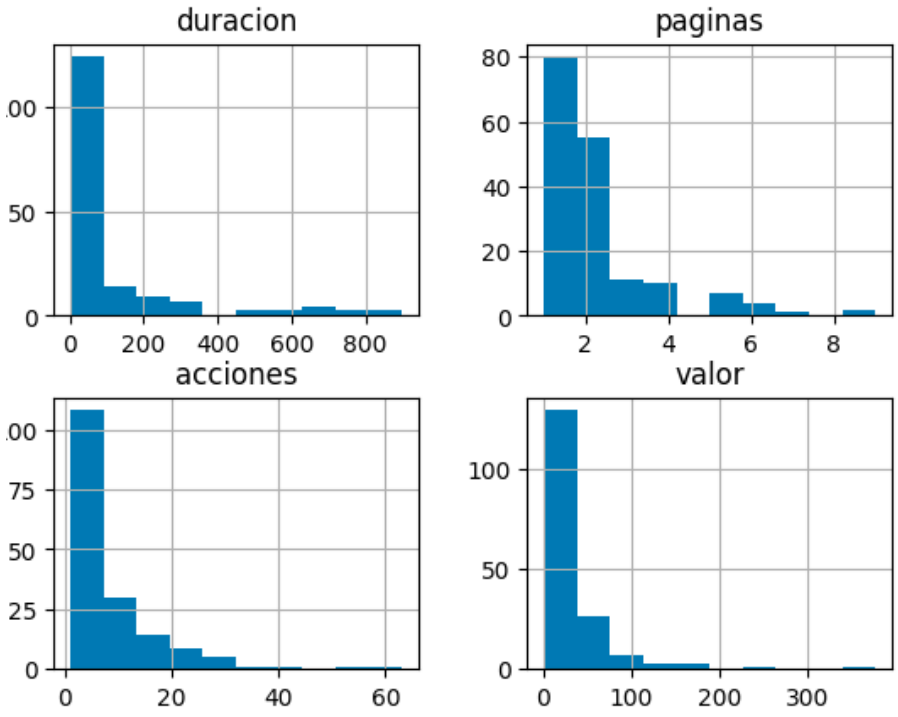


Figure 4: Histogramas

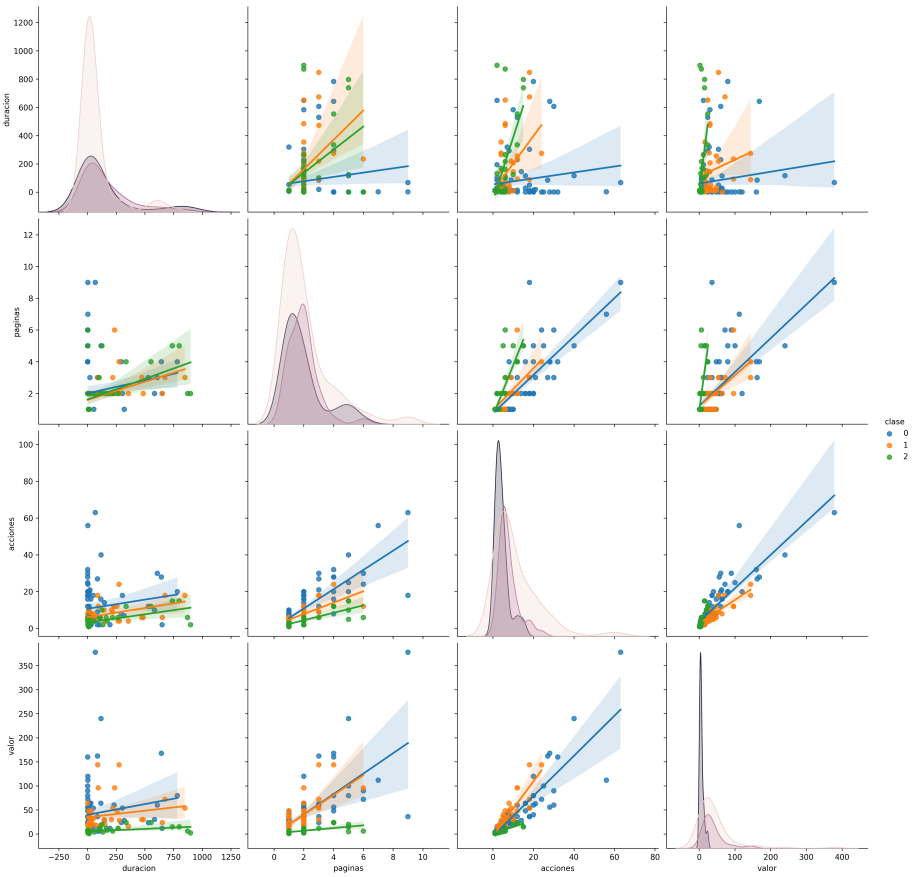


Figure 5: Pairplot

## 2.3 Entrenamiento del modelo

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

model = LogisticRegression(max_iter=1000)
model.fit(X, y)
print(model.score(X, y))
```

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

]: (170, 4)
```

Figure 6: Modelo

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

: 0.7764705882352941
```

Figure 7: Score Modelo

## 2.4 Evaluación del modelo

```
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)

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')
print("Regresi\'on Log\'istica: %f (%f)" % (cv_results.
    mean(), cv_results.std()))
```

```
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: 0.720350 (0.151123)
```

Figure 8: Validación

## 2.5 Predicción y métricas

```
predictions = model.predict(X_validation)
print("Precisi\'on:", accuracy_score(Y_validation,
    predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

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

0.8529411764705882
```

Figure 9: Predicciones

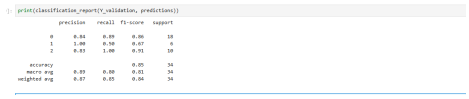


Figure 10: Clasificación

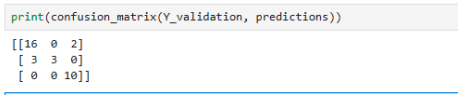


Figure 11:

### 2.6 Predicción de nuevos datos

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


Figure 12: Predicciones

## 3 Resultados

El modelo de Regresión Logística desarrollado para clasificar el sistema operativo de los usuarios ha sido evaluado mediante la métrica `classification_report`, que proporciona información sobre la precisión, recall y F1-score para cada clase. A continuación, se interpretan los resultados obtenidos.

### 3.1 Desempeño General del Modelo

El modelo logró una precisión global del 85%, lo que indica que, en promedio, clasifica correctamente el 85% de los casos en el conjunto de validación. Esta es una precisión relativamente alta, lo que sugiere que el modelo es eficaz en la tarea de clasificación.

### 3.2 Análisis por Clase

Se han identificado diferencias en el desempeño según la clase:

- **Clase 0 (Windows)**:

  - **Precisión**: 84%
  - **Recall**: 89%
  - **F1-score**: 86%
  - **Interpretación**: El modelo identifica la mayoría de los casos de Windows correctamente y rara vez los clasifica erróneamente.

- **Clase 1 (Macintosh)**:
  - **Precisión**: 100%
  - **Recall**: 50%
  - **F1-score**: 67%
  - **Interpretación**: Aunque el modelo nunca se equivoca cuando predice un caso como Macintosh (100% de precisión), solo identifica la mitad de los casos reales de esta clase (50% de recall). Esto sugiere que el modelo es demasiado conservador al clasificar un dato como Macintosh y podría estar subrepresentando esta clase.

- **Clase 2 (Linux)**:
  - **Precisión**: 83%
  - **Recall**: 100%
  - **F1-score**: 91%
  - **Interpretación**: El modelo detecta todos los casos reales de Linux (recall del 100%), pero en algunas ocasiones clasifica erróneamente otros datos como si fueran de esta categoría (83% de precisión).

# 4 Conclusión

Se logró entrenar y evaluar un modelo de regresión logística para clasificar usuarios según su sistema operativo basado en su actividad en el sitio web. El modelo demostró un buen desempeño general, alcanzando una precisión del 85%, aunque se observaron ciertas limitaciones en la clasificación de la clase Macintosh debido a un bajo recall.

Para mejorar el rendimiento del modelo, se recomienda:

- Aumentar la cantidad de datos de entrenamiento para la clase Macintosh.
- Ajustar los hiperparámetros para equilibrar la clasificación entre las distintas clases.
- Probar otros modelos de clasificación como SVM o Random Forest para comparar resultados.

Con estos ajustes, se podría obtener un modelo más robusto y preciso para la clasificación del sistema operativo de los usuarios.