KNN, REGRESIÓN LINEAL Y NAIVE BAYES

WIN-LOSE PREDICTOR

- Andrés Jaramillo Barón - A01029079
- Pedro Mauri Martínez - A01029143
- Ricardo Calvo Pérez - A01028889

AGENDA

CONTEXTO DE LA PROBLEMÁTICA

Predicción de resultado (ganar o perder) de partida de videojuegos, dependiendo de las circunstancias actuales del jugador.

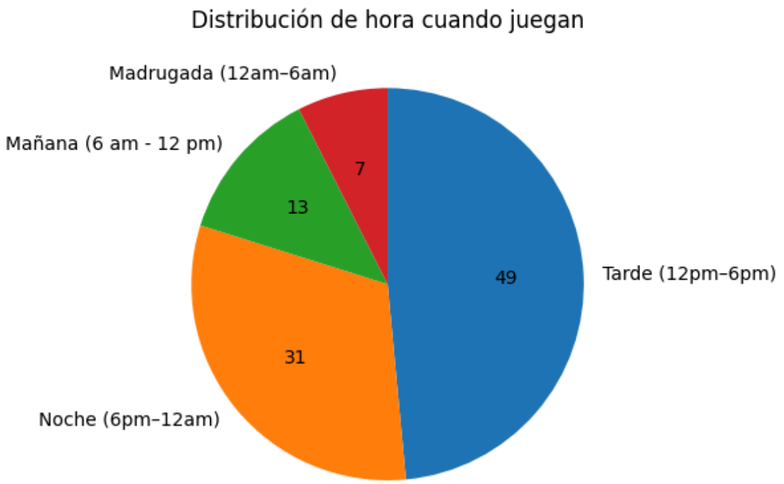
¿CÓMO SE OBTUVO EL DATASET?

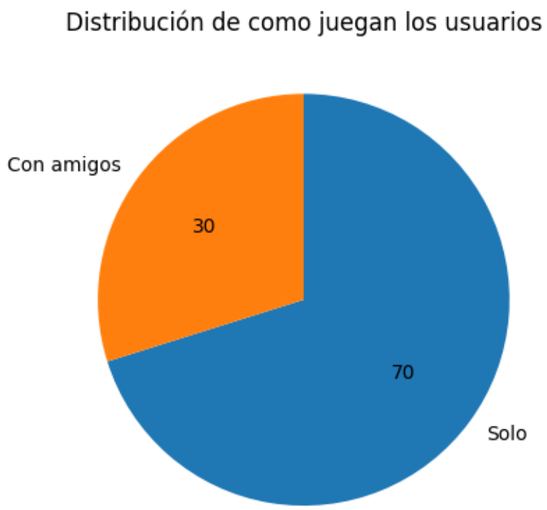
- Se creó un google forms para recolección de datos.
- Se compartió con amigos, en grupos de discord.
- Al registrar una partida, el usuario puede enviar varias respuestas gracias al historial
- Muestra rápida del dataset

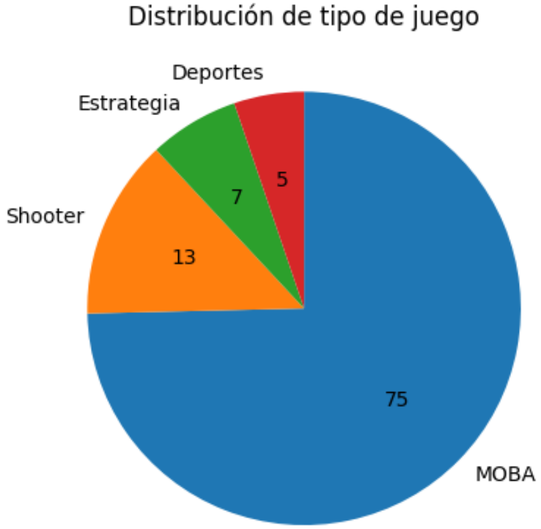
	Horas jugadas antes de la partida	Hora del día en que jugaste	Tipo de juego	Jugaste con amigos o solo	Nivel de cansancio antes de jugar	Número de victorias consecutivas que llevabas antes de esta partida	Ganaste o Perdiste la partida
0	4 - 5 + horas	Madrugada (12am–6am)	Deportes	Solo	4	10	Perdí
1	0 - 1 horas	Noche (6pm–12am)	Shooter	Con amigos	4	3	Gané
2	0 - 1 horas	Noche (6pm–12am)	MOBA	Con amigos	3	0	Perdí
3	2 - 3 horas	Tarde (12pm–6pm)	MOBA	Solo	4	3	Gané
4	1 - 2 horas	Mañana (6 am - 12 pm)	MOBA	Solo	1	0	Gané

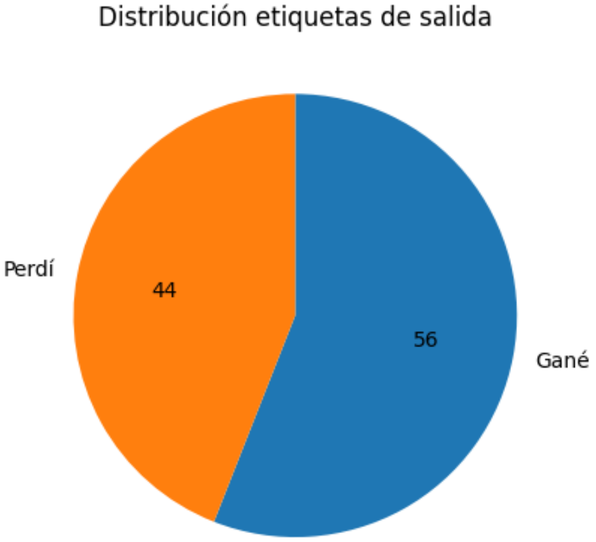
ANÁLISIS DE DATASET

Se creó un notebook .ipynb para realizar el análisis del dataset, lo cual nos permitió comprender el comportamiento de los modelos con mayor facilidad. +130 muestras.





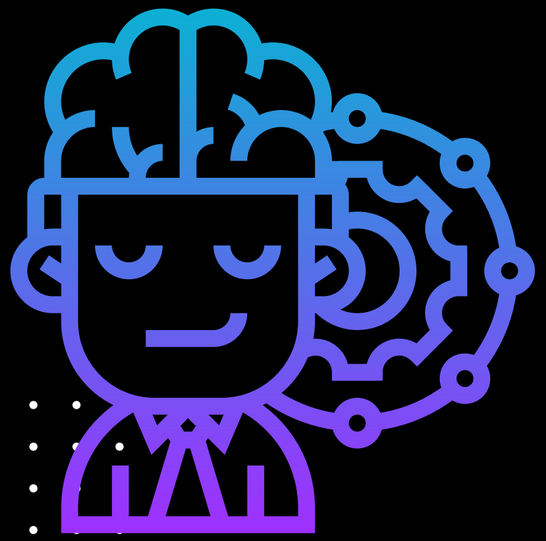




MODELOS DE MACHINE LEARNING

Se implementaron los siguientes algoritmos de clasificación binaria:

- KNN
- Regresión logistica
- Naive Bayes

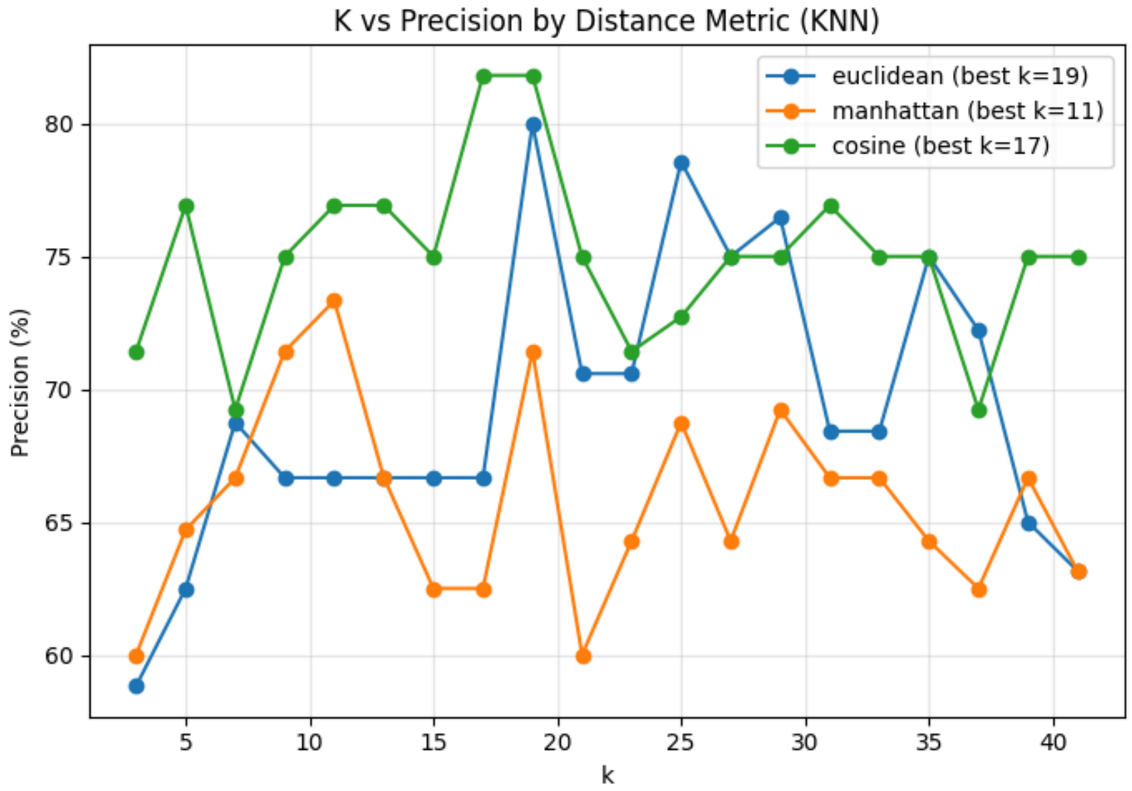


Se utilizó la librería de Scickit-learn para las 3 implementaciones y se calcularon las siguientes métricas para cada modelo:

- Accuracy
- Precision
- Recall
- F1
- ROC / AUC

RENDIMIENTO

Métricas de rendimiento de los modelos y análisis de knn (valor de k y distancia usada)




```
============================== KNN ==============================
--- Applying KNN using scikit-learn ---
Training KNN...
Testing KNN...
--- KNN Metrics (k = 17 | cosine | Weights = uniform) ---
Model accuracy: 0.7037037037037037
Model precision: 0.8181818181818182
Model recall: 0.6
Model F1 score: 0.6923076923076923
Model ROC / AUC: 0.7166666666666668
Model saved to backend/artifacts/knn_model.joblib

============================== Logistic Regression ==============================
--- Applying Logistic Regression using scikit-learn ---
Training Logistic Regression...
Testing Logistic Regression...
--- Logistic Regression Metrics (lambda = 1) | Ridge (L2) ---
Model accuracy: 0.6296296296296297
Model precision: 0.6923076923076923
Model recall: 0.6
Model F1 score: 0.6428571428571429
Model ROC / AUC: 0.6333333333333334
Model saved to backend/artifacts/logreg_model.joblib

============================== Naive Bayes ==============================
--- Applying Naive Bayes using scikit-learn ---
Training Naive Bayes...
Testing Naive Bayes...
--- Naive Bayes Metrics ---
Model accuracy: 0.5185185185185185
Model precision: 0.5833333333333334
Model recall: 0.4666666666666667
Model F1 score: 0.5185185185185185
Model ROC / AUC: 0.525
Model saved to backend/artifacts/nb_model.joblib
```

BACKEND

Lectura, análisis y preparación del dataset

Desarrollo de los 3 modelos de ML

Script para intercambio de datos entre backend y frontend

Serialización de los modelos para realizar pruebas con datos nuevos (guardar entrenamiento)

FRONTEND

Cuestionario para recopilar los datos del usuario

Script de intercambio de datos entre backend y frontend

Muestra de resultados de la prueba

MUESTRA DE PROYECTO

¡Gracias por su atención!