Facultad de Ingeniería y Ciencias
Escuela de Informática y Telecomunicaciones

Tarea 2

Inteligencia Artificial

CIT-3346

Nicolás Flores Sepúlveda

Profesor: Martín Gutiérrez

Ayudante: Javier Carrión

Junio, 2019

1. Introducción

En esta tarea se pidió crear y comparar dos algoritmos clasificadores/predictores. En los cuales se debe evaluar la eficiencia y efectividad de ambos.

En una primera instancia se buscó un dataset acorde para poder clasificar y predecir los datos con los algoritmos creados.

Luego se utilizó un algoritmo de clasificación para agrupar los datos del dataset. Posteriormente se utilizó un algoritmo de predicción, para saber en qué grupo se asignara un nuevo dato.

En la segunda parte de la tarea se utilizó un mismo algoritmo para realizar los dos pasos anteriores, esto es clasificar y predecir.

Finalmente se compararon los algoritmos implementados, midiendo sus

tiempos de ejecución, cuanta memoria ram utilizan y que tan precisos son al realizar la clasificación y predicción.

2. Desarrollo

2.1. DataSet

Para realizar esta tarea se eligió un dataset que se encuentra en la página web *kaggle.com*.

El dataset seleccionado contiene datos sobre algunas partidas competitivas del videojuego *League of Legends*, en donde se especifican los siguientes datos:

1. Id de la partida
2. Duración de la partida
3. Id de la temporada
4. Ganador (1 = team1, 2 = team2)
5. Primer Barón, dragón, inhibidor, Heraldo, torre eliminada y primera sangre realizada (1 = team 1, 2 = team2, 0 = ninguno)
6. Campeón y hechizo elegido por cada jugador (Asignadas según la ID de los campeones y hechizos definidos por Riot)
7. Número de torres, inhibidores, Barón, dragón y Heraldos eliminados por cada equipo.
8. Los 5 bloqueos de campeones realizados por cada equipo.

Luego de seleccionar el dataset, se creó un gráfico de 3 dimensiones, en donde los ejes del plano tridimensional fueron los siguientes:

- Cantidad de inhibidores destruidos por el equipo 1.
- Cantidad de barones eliminados por el equipo 1
- Cantidad de torres destruidas por el equipo 1.

Estos datos fueron seleccionados debido a que para ganar una partida se deben destruir algunas torres e inhibidores enemigos, por lo que al destruir una mayor cantidad de estos objetivos la probabilidad de ganar una partida aumenta. Por otro lado el Barón es un monstruo poderoso que otorga un bonus de daño al equipo que lo mata.

Además, se asignó un color a cada partida según el equipo ganador. El color Azul indica que el equipo 1 gano la partida. En cambio el color Rojo indica que el equipo 2 fue el vencedor.

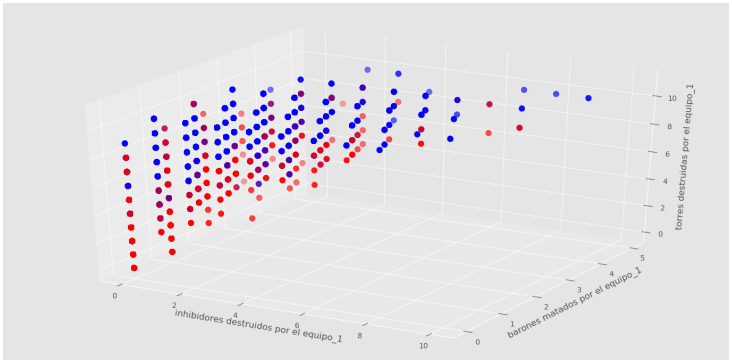


Figura 1: Graficos del dataset con 3 variables

Como acotación se puede mencionar que los datos seleccionados tal vez no sean significativos para decidir si el resultado de la etiqueta elegida, por lo que si se eligen otros datos se puede obtener una mayor precisión.

2.2. Primera Parte

2.2.1. Clustering

Para realizar el clustering se utilizó K-means sobre los datos, con K=2, para que de esta forma se creen dos grupos. En un grupo estarán las partidas que gano el equipo 1, el cual se destacara con el color Azul y en el otro estarán las partidas que gano el equipo 2 (Rojo).

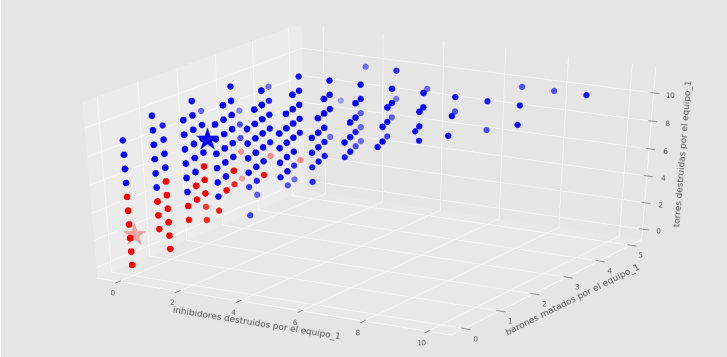


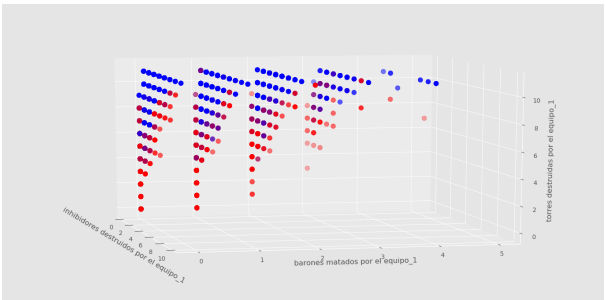
Figura 2: Grafico al utilizar K means

Los centroides de cada grupo se ubican en las posiciones :

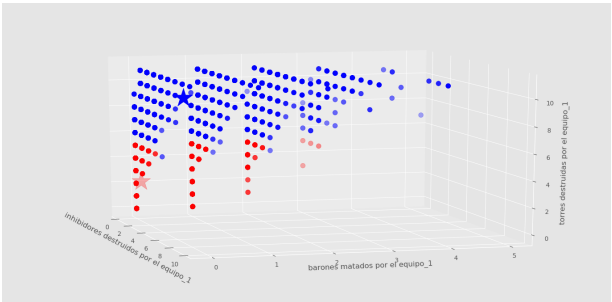
- Azul: (1.92156278, 0.63432001 , 8.97696093)
- Rojo: (0.03441985, 0.08732668, 2.13500365)

De la agrupación realizada por el algoritmo K-Means, se puede comprobar que entre más objetivos destruya o elimine el equipo 1 (Torres, Inhibidores y Barones), es más probable que el equipo 1 gane la partida.

Por esta razón el grupo Azul se encuentra en la parte superior del gráfico. A diferencia del grupo rojo que se encuentra en la parte inferior, el cual indica que el equipo 2 gano la partida.



(a) Gráfico del dataset



(b) Grafico al realizar K means

Figura 3: Comparación entre los graficos

2.3. Predicción

El algoritmo que se utilizó para predecir fue Regresión logística, debido a que las etiquetas seleccionadas son parámetros discretos.

Para entrenar el algoritmo de predicción se utilizó las etiquetas entregadas por K-Means.

2.4. Segunda Parte

En la segunda parte de la tarea se utilizó el algoritmo KNN, el cual permite clasificar y predecir.

2.4.1. Clasificación

Para poder clasificar los datos con KNN se debe separar los datos del dataset en dos. Por un lado deben estar los datos y las etiquetas para entrenar el algoritmo y por el otro lado deben estar los datos y etiquetas para probar si el algoritmo es preciso. Para realizar este paso se utilizó la función train-test-split de la librería sklearn, la divide al azar los datos de entrenamiento y de prueba.

Posteriormente se debe elegir la cantidad de vecinos que analizará KNN, en esta ocasión se eligió revisar 4 vecinos. Luego se entrenó el algoritmo con los datos y etiquetas de entrenamiento para poder clasificar los datos de prueba utilizando la función predict de KNN.

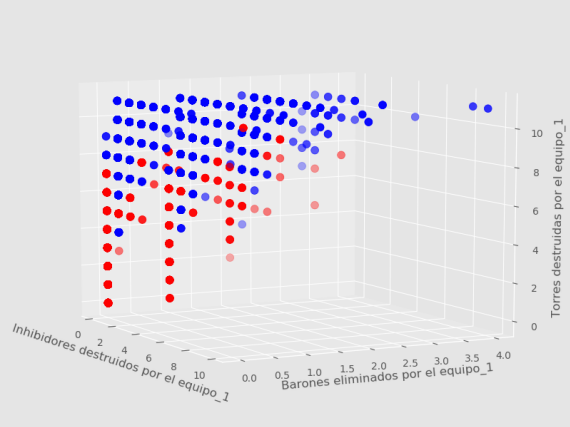
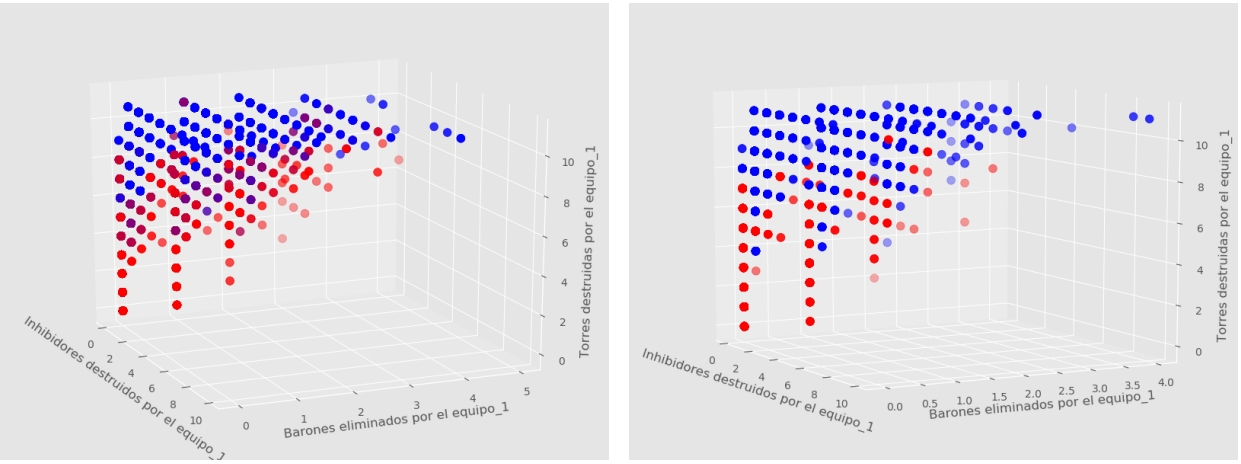


Figura 4: Agrupación realizada por KNN

La agrupación realizada es similar a la agrupación real de los datos del dataset.

Esto se puede visualizar en la Figura 5. Cabe destacar que se aprecian menos puntos en el gráfico de KNN debido a que solo se gráfica la porción de los datos que corresponde a los datos de prueba.



(a) Gráfico del dataset (b) Gráfico al realizar KNN

Figura 5: Comparación entre la agrupación del dataset y KNN

2.4.2. Predicción

Para realizar la predicción se eligieron datos al azar y se comprobó el grupo que le asignaba el algoritmo KNN al utilizar la función predict.

2.5. Comparación entre los algoritmos utilizados

2.5.1. Eficiencia:

1. El algoritmo creado en la primera parte de la tarea se demorí 0.877 unidades de tiempo en ejecutarse y utilizo 109.49 Mb de memoria ram.

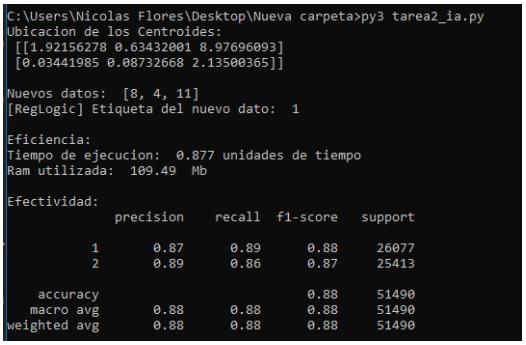
2. El algoritmo de la segunda parte se demoró 2.133 unidades de tiempo y utilizo 109.51 Mb de memoria ram

Se debe mencionar que estas mediciones se realizaron sin realizar los gráficos para poder analizar el desempeño real de los algoritmos utilizados.

2.5.2. Efectividad:

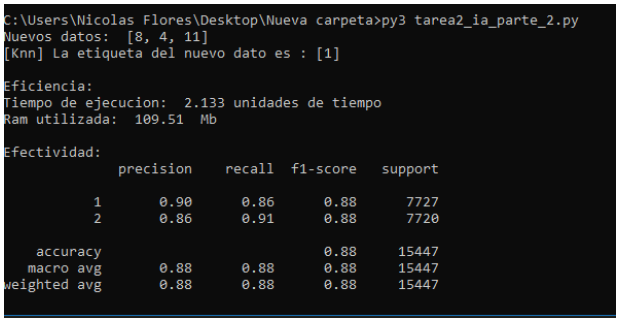
1. La precisión del algoritmo creado en la primera parte fue del 88 %.
2. La precisión del segundo algoritmo fue de 88 %.

Cabe destacar que la precisión del algoritmo KNN depende de los parámetros de entrenamiento y prueba seleccionados, es decir que si se cambian los valores la precisión puede aumentar o disminuir.



```
C:\Users\Nicolas Flores\Desktop\Nueva carpeta>py3 tarea2_ia.py
Ubicacion de los Centroides:
 [[1.92156278 0.63432001 8.97696093]
 [0.03441985 0.08732668 2.13500365]]

Nuevos datos:  [8, 4, 11]
[RegLogic] Etiqueta del nuevo dato:  1

Eficiencia:
Tiempo de ejecucion:  0.877 unidades de tiempo
Ram utilizada:  109.49  Mb

Efectividad:
              precision    recall  f1-score   support

           1       0.87      0.89      0.88     26077
           2       0.89      0.86      0.87     25413

    accuracy                           0.88     51490
   macro avg       0.88      0.88      0.88     51490
weighted avg       0.88      0.88      0.88     51490
```

(a) Output del primer algoritmo



```
C:\Users\Nicolas Flores\Desktop\Nueva carpeta>py3 tarea2_ia_parte_2.py
Nuevos datos:  [8, 4, 11]
[Knn] La etiqueta del nuevo dato es : [1]

Eficiencia:
Tiempo de ejecucion:  2.133 unidades de tiempo
Ram utilizada:  109.51  Mb

Efectividad:
              precision    recall  f1-score   support

           1       0.90      0.86      0.88      7727
           2       0.86      0.91      0.88      7720

    accuracy                           0.88     15447
   macro avg       0.88      0.88      0.88     15447
weighted avg       0.88      0.88      0.88     15447
```

(b) Output del segundo algoritmo

Figura 6: Comparación entre K-Means con Regresión logística y KNN.

2.6. Observaciones del Algoritmo

El algoritmo esta escrito en Python 3.5.4 y para que funcione se debe tener instalado las siguientes librerias: pandas, numpy, matplotlib, (KMeans, pairwise-distances-argmin-min, Axes3D,LogisticRegression,linear-model,mean-squared-error, r2-score, train-test-split, StandardScaler, KNeighborsClassifier, classification-report, confusion-matrix), os, psutil y time.

3. Conclusión

De esta experiencia no se puede concluir que algoritmo es más efectivo a la hora de realizar predicción y agrupaciones, ya que ambos entregaron el mismo porcentaje de precisión, pero si se puede mencionar que el algoritmo de la primera parte en donde se utiliza K-Means y regresión logística utiliza menos recursos y se demora menos tiempo en ejecutarse. Esto de puede deber

a que con KNN se deben separar los en dos, en datos de entran amiento y prueba.

Se mencionar que se puede realizar un experimento a futuro en donde podría cambiar los datos elegidos, utilizando la misma etiqueta y analizar los resultados de ambos algoritmos creados.

4. Referencias

https://www.kaggle.com/datasnaek/league-of-legends

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

https://scikit-learn.org/stable/modules/generated/sklearn.model-selection.train-test-split.html