**A close up of a sign

Description automatically generated**

**Proyecto Final**EMI Music Para Predecir Ratings

Instructor: Dr. Alejandro Rosales P

Layla Tame A01192934

2 de Diciembre, 2019

horizontal line

**Resumen**

Para la realización de este proyecto se pretende hacer uso de la inteligencia computacional a través de la implementación de diversos algoritmos para logar una correcta predicción sobre una serie de datos. La aplicación del reporte va enfocado hacia la predicción de ratings de nuevas canciones musicales tomando en cuenta los gustos pasados del usuario. Se trabajará con datos de los diversos usuarios, artistas y canciones para poder predecir su una nueva canción sería del agrado de una persona en especifico.

Se implementarán cuatro diferentes algoritmos de clasificación pertenecientes a la rama de la inteligencia computacional, siendo en este caso una máquina de soporte vectorial, un algoritmo multiobjetivo, una función KNN y finalmente árboles de decisión mejor conocidos como “Random Forest”.

Se pretende analizar los datos obtenidos de las predicciones de los diferentes algoritmos y compararlos entre sí para encontrar cual de ellos se acerca de mejor manera a los datos reales, dando así la solución más precisa. Para ello se hará uso de dos librerías de Python: sklearn y jMetal.

Finalmente se pretende encontrar las posibles áreas de mejora de los algoritmos implementados para, en un futuro, poder crear una versión mejorada que provea resultados más precisos y lo haga de la manera más eficiente posible.

**Introducción**

A lo largo de los últimos años, el campo de la inteligencia artificial ha crecido de manera exponencial, y con ella la inteligencia computacional. Estas ramas han permitido que se desarrollen algoritmos y productos que facilitan los trabajos normalmente realizados por un ser humano; y no solo eso, si no que brindan resultados más precisos y de manera más rápida. Se ha logrado aplicar la IA en una variedad de industrias, desde la medicina hasta la industria manufacturera, le implementación de diversos algoritmos ha tenido resultados impresionantes.

La inteligencia computacional, rama de lo que conocemos como inteligencia artificial, pretende incorporar procesos de aprendizaje y adaptación a los dispositivos electrónicos. Esta permite buscar resultados a través de la implementación de diversas estrategias, utilizando redes neuronales, algoritmos evolutivos, entre otros, adaptándolos según las necesidades del problema que se desea optimizar o resolver. Uno de los principales propósitos de la inteligencia computacional es permitir la clasificación de datos a ciertas categorías con el fin de poder mejorar procesas, aumentar conocimientos sobre cierta área, etc.

Para el desarrollo de este propósito, se decidió aplicar la inteligencia computacional a la industria de la música. Se analizó una serie de datos con información acerca de usuarios, las canciones y artistas que escuchan y el rating que le dan a cada una de ellas. El objetivo es obtener suficiente información a través del uso de una máquina de soporte vectorial y un algoritmo multiobjetivo para poder predecir qué tanto le gustaría una canción a un usuario, tomando como base para entrenar el algoritmo su historial pasado. De esta manera, se toman en cuenta los gustos del usuario, que artistas son de su interés y el parecido entre esos artistas a otros para poder determinar si la nueva canción le gustará. Se utilizarán diversos algoritmos de clasificación para comparar los diferentes resultados obtenidos y poder obtener el mejor para los datos.

A lo largo del documento se presentará el data set para tener una idea completa de los parámetros en cuestión. Se presentarán las tecnologías y librerías que se utilizaron para facilitar la implementación de los algoritmos, así como los resultados que se obtuvieron con cada uno de ellos. Se realizará una comparación entre los resultados para poder determinar cual fue la mejor solución.

**Materiales y Métodos**

Tras analizar diferentes data sets que existen en internet, se eligió uno llamado “EMI Music” el cual contiene información acerca de diversos usuarios, el estilo de música que les gusta escuchar, así como las calificaciones que le han dado a ciertas canciones y artistas con base en sus gustos musicales. Los principales archivos que componen el data set son los siguientes:

* Train/Test

Este archivo contiene información mostrando el número que identifica a cada usuario, el número que identifica a cada artista y el número que identifica a cada canción. Se muestra también una variable de tiempo que representa los momentos que pasa el usuario escuchando a su música, así como una columna donde se muestra la calificación que el usuario asignó a cada canción en un número entero entre el 1 y el 100.

* Words

El archivo muestra datos acerca de cómo un usuario describe el tipo de música que escucha; incluye palabras que indican sentimientos que tuvo el usuario con cada una de las canciones, si alguna vez había escuchado del artista, si le gusta o no escucharlo, así como sus gustos musicales específicos.

* Userkey/Users

Información acerca del usuario. Esta incluye su género, edad, estatus laboral, región en donde vive, que tan importante considera la música como parte de su vida, cantidad de tiempo que pasa escuchando música, así como sus hábitos respecto al tema.

Para propósitos de este proyecto, se decidió omitir el uso del archivo que contiene información detallada acerca de las canciones (Words) ya que implicaba realizar un análisis mucho más extenso utilizando variables cualitativas. Lo mismo para los archivos de Userkey y Users. Por el tipo de algoritmos seleccionados, se decidió trabajar usando variables cuantitativas, las que respectan a la calificación asignada, el número de usuario, número de canción y tiempo que pasa escuchando música. El archivo utilizado como datos de entrada contiene más de 180,000 entradas con datos de 49 artistas, 183 canciones y 50,927 usuarios. A continuación de muestra la distribución de los tres principales datos.

A picture containing fence

Description automatically generated A picture containing chair, fence, drawing

Description automatically generated A picture containing fence

Description automatically generated

Artista Canciones Usuarios

Para el desarrollo del proyecto, se investigó acerca de los diferentes algoritmos de clasificación que ya se conocen y se han implementado con la intención de descubrir cuales podrían ser aplicados a los datos que se tenían para analizar. Tras la investigación se decidió implementar cuatro algoritmos: máquina de soporte vectorial, algoritmo multiobjetivo utilizando un problema binario, el algoritmo KNN y arboles de decisión, específicamente un algoritmo llamado “Random Forest”.

Los algoritmos fueron implementados utilizando Python, específicamente las librerías de Scikit-Learn y jMetal. Scikit-Learn es una librería de machine learning que facilita herramientas para el análisis de datos y contiene la implementación de algoritmos de clasificación, regresión, clustering, reducción de dimensiones, preprocesamiento de datos, selección de modelos, entre otros. De esta se obtuvieron para su implementación los algoritmos de KNN, máquina de soporte vectorial (SVC) y Random Forest. Por otro lado, jMetal es una librería que contiene implementaciones de algoritmos evolutivos. En este caso se utilizó el algoritmo NSGAII, así como los operadores mutación BipFlit y de cruza SPX.

Cómo métricas a utilizar para realizar el análisis de la eficiencia y precisión de cada uno de los algoritmos, se utilizaron las métricas disponibles en SKLearn para calcular una predicción conforme a unos datos de entrada, la exactitud de los datos predichos contra los reales, así como diversas métricas de clasificación.

Se tuvo que realizar una limpia de los datos para lograr un entrenamiento lo más preciso posible. El archivo utilizado para entrenar muestra la siguiente distribución:

A screenshot of a cell phone

Description automatically generated

Posteriormente se discretizaron los datos de rating, dividiendo en cuatro categorías enumeradas del 1 al 4. Todas las entradas con una calificación menor a 25 se establecieron como categoría 1, todas aquellas entre 25 y 50 se calificaron como un 2, aquellas entre 50 y 75 como un 3 y las calificaciones mayores a 75 con un 4. Esto con el propósito de que el entrenamiento y la clasificación fuera más rápida y sencilla.

**Resultados**

Para la evaluación de los algoritmos se generaron dos archivos de entrada. El primero, X, contiene la información de los artistas, usuarios, canciones y tiempos. El segundo archivo, y, contiene los datos de calificaciones asignadas a cada una de las entradas. Se utilizó la función de train\_test\_split para generar cuatro archivos que serían utilizados para el entrenamiento y las pruebas con cada uno de los algoritmos: X\_train, X\_test, y\_train y y\_test. Se determinó que el 75% de los datos serían utilizados para el entrenamiento y el 25% restante para realizar las pruebas.

El primer algoritmo que se implementó fue una máquina de soporte vectorial. Se inicializaron los parámetros con los siguientes valores:

A screenshot of a cell phone

Description automatically generated

El algoritmo se probó en múltiples ocasiones utilizando diferentes valores de la función kernel para encontrar aquel que diera mejores resultados. Se utilizó la función lineal, sigmoid, polinomial y Gausiana (rbf). Se mandó llamar la función SVC implementada a través de sklearn con los valores establecidos y utilizando los arreglos de datos X\_train y y\_train. Los resultados obtenidos fueron los siguientes:

A close up of a piece of paper

Description automatically generated A close up of a piece of paper

Description automatically generated

Sigmoid Lineal

A close up of a piece of paper

Description automatically generated A close up of a piece of paper

Description automatically generated

Polinomial Gausiana

La implementación con mayor exactitud fue en la que se utilizó la función sigmoid de kernel con un resultado del 52%. Las otras tres funciones implementadas dieron resultados mucho menos precisos con más de 10% de diferencia entre ellos y sigmoid.

Se realizó el mismo procedimiento para la implementación del algoritmo multiobjetivo. Para este se decidió utilizar un problema binario y el algoritmo NSGAII disponible a través de la librería de jMetal. La inicialización de los parámetros utilizados para entrenar fue la misma que los usados en la máquina de soporte vectorial. De la misma manera, se probó el algoritmo utilizando las 4 diferentes funciones de kernel; los resultados fueron los siguientes:

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

Sigmoid Gausiana

De nuevo, el resultado más preciso fue el obtenido a través de la función Sigmoid con una exactitud del 44.4%. Se puede observar a través del reporte de clasificación que la categoría con mejor clasificación fue la 1, seguida por la categoría 2. Las funciones de kernel Lineal y Polinomial fueron descartadas como alternativas prometedoras ya que los resultados obtenidos no solamente eran mucho menos precisos a los otros dos, si no que el tiempo de ejecución y de entrenamiento del algoritmo era muchísimo mayor, sobre todo tratándose de un data set muy grande.

El tercer algoritmo implementado fue el KNN, también conocido como “K vecinos más cercanos”. Este algoritmo es utilizado frecuentemente para diversas aplicaciones en el mundo de machine learning. El algoritmo es relativamente sencillo de implementar y tiene como principal objetivo obtener aquellas instancias más “similares” o “cercanas” entre sí en cada etapa del entrenamiento con la intención de crear nuevos puntos.

Para este algoritmo se itero una vez por cada uno de los datos de entrada en el archivo de y\_train. Como resultado se obtuvo un arreglo que contiene las diferentes precisiones conforme a los datos de entrada y su relación con los datos reales. Tras la ejecución del mismo se obtuvo que el puntaje más alto de relación o exactitud de los datos fue de 42.8%.

A close up of a logo

Description automatically generated

Finalmente se puso en práctica el algoritmo de “Random Forest” el cual representa una versión de arboles de decisión un poco más complejo y avanzado. Este consiste en un número grande de arboles individuales de decisión que trabajan en conjunto para realizar clasificaciones. El resultado obtenido fue el siguiente:

A close up of a logo

Description automatically generated

Se puede observar que la exactitud de la predicción contra los datos reales fue relativamente baja, con un porcentaje de similitud del 34.4%.

**Conclusión**

Tras realizar la limpia y el análisis de los datos para su procesamiento, se pudo determinar los algoritmos a implementar para poder encontrar el más preciso. La intención de implementar más de uno fue poder comparar su rapidez, exactitud y actuación en general. Al observar los resultados obtenidos se puede determinar que el algoritmo con mejor resultado fue la implementación de la máquina de soporte vectorial a través de la librería de sklearn, seguido por la implementación del algoritmo multiobjetivo, ambos siendo aplicados utilizando la función sigmoid de kernel. El primer resultado muestra una correlación y exactitud del 52%, indicando que el 52% de las predicciones realizadas fueron correctas contra los datos reales.

Los resultados obtenidos tras la implementación de las diferentes soluciones no fueron óptimos, ya que las predicciones pudieron haber sido mucho más precisas. Se llegó a la conclusión de que los resultados pueden ser mejorados significativamente al entrenar los algoritmos utilizando la población entera de datos de entrada que contiene el set. Para propósitos de este proyecto, se tomó una muestra, la cual se intentó fuera representativa de la población total, ya que el archivo de entrada contenía más de 180,000 datos, lo cual hacía que la ejecución del algoritmo fuera muy lenta debido a la capacidad de procesamiento de la computadora que se utilizó. Se notó que conforme incrementan los datos utilizados para el entrenamiento, mucho mejor es la actuación de los diferentes algoritmos. Debido a que las predicciones se realizan tomando en cuenta los gustos de cada usuario, basándose en información ya conocida sobre los artistas y las canciones que le han gustado en el pasado y cuales no, mientras más información se provee acerca de los diferentes usuarios, mejores resultados se pueden obtener.

Cómo mejora para el proyecto en un futuro, se propone encontrar la manera de incluir los archivos “Users” y “Words” en el análisis de los datos y el entrenamiento de los algoritmos. Aun que esto involucra un algoritmo y una implementación mucho más compleja a la aprendida durante este curso, estos archivos contienen información que permite realizar un reporte mucho más completo, tomando en cuenta los gustos particulares de un usuario, qué tan expresivo es respecto a la música que utiliza para dar diferentes ponderaciones a sus opiniones, tomar en cuenta la edad o su conocimiento sobre la música, entre otros. Otras posibles soluciones para mejorar la actuación del algoritmo incluyen jugar con los parámetros de inicialización de manera dinámica para poder encontrar aquellos que provean una predicción más precisa, o incluso varear las implementaciones de cruza y mutación que fueron utilizadas.

**Referencias**

* Bhanot, K. (2019, February 12). Predicting presence of Heart Diseases using Machine Learning. Retrieved from <https://towardsdatascience.com/predicting-presence-of-heart-diseases-using-machine-learning-36f00f3edb2c>.
* Documentation of scikit-learn 0.21.3. (n.d.). Retrieved from https://scikit-learn.org/stable/documentation.html.
* Documentation of jMetal. (2019, November 6). jMetal/jMetalPy. Retrieved from https://github.com/jMetal/jMetalPy.
* Malik, U. (2019, May 9). Implementing SVM and Kernel SVM with Python's Scikit-Learn. Retrieved from <https://stackabuse.com/implementing-svm-and-kernel-svm-with-pythons-scikit-learn/>.
* Na8. (2019, May 21). Clasificar con K-Nearest-Neighbor ejemplo en Python. Retrieved from https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/.
* Rodrigo, J. A. (2017, April). Máquinas de Vector Soporte (Support Vector Machines, SVMs). Retrieved from https://rpubs.com/Joaquin\_AR/267926.
* Van Vaerenbergh, S., & Santamaría, I. (2018, March 20). PDF. Retrieved from <https://gtas.unican.es/files/docencia/APS/apuntes/07_svm_kernel.pdf>
* Yiu, T. (2019, August 14). Understanding Random Forest. Retrieved from https://towardsdatascience.com/understanding-random-forest-58381e0602d2.