Universidad Nacional
Autónoma de México

Inteligencia Artificial

Proyecto Final: Jarreth

Grupo 3

Nombre:
Barreiro Valdez Alejandro

Proyecto

Profesor: Dr. Guillermo Gilberto Molero Castillo

26 de mayo de 2022

1. Introducción

Jarreth es un programa hecho con un *framework* de *Python* llamado *Dash*, capaz de resolver problemas de inteligencia artificial, específicamente de aprendizaje automático, utilizando datos que proporciona el usuario. Jarreth es ideal como introducción a los algoritmos más utilizados en el aprendizaje automático. Permite al usuario modificar parámetros de cada uno de los modelos y entender cada uno de los algoritmos con la información que contiene.

1.1. Requerimientos

Se utilizó Python 3.6.8 para el desarrollo de este código. Adicionalmente, para poder correr el programa se necesitan tener muchos módulos útiles para le generación de modelos de aprendizaje automático. Se recomienda tener Anaconda instalado y utilizando el comando *pip* instalar los siguientes paquetes: *dash*, *dash-bootstrap-components* y *apyori*. También se utilizan múltiples paquetes de *sklearn*.

1.2. Correr el programa

El código se ejecutó con esta versión de Python utilizando el comando *python3*. Se utiliza dicho comando y se ejecuta el archivo de *index.py*. Posteriormente, el programa debe desplegar la información del *url* al que se debe ingresar para visualizar el programa. La primera pantalla que debe aparecer debe ser la de la Figura 1.

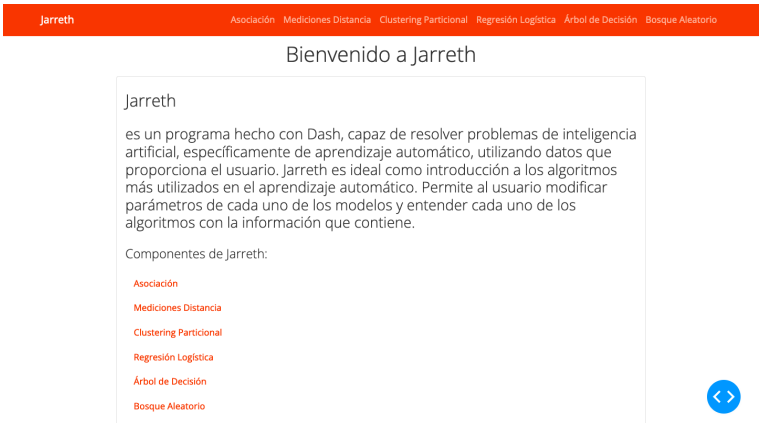

Figura 1: Bienvenida a Jarreth

1.3. Subir datos

Para subir datos se debe utilizar la sección que cada componente provee para ello. Los datos que se suben deben ser con el formato *.csv*. Estos datos no se pueden modificar dentro de los componentes. El único cambio que se hace es la conversión de cadenas a datos de tipos numéricos a excepción del componente de reglas de asociación. Se debe verificar que los datos que se suban sean válidos, de lo contrario el componente no ejecutará lo que se pide. Se debe trabajar con los componentes hasta que se haya subido un archivo.

2. Componentes de la aplicación

La aplicación tiene siete componentes: reglas de recomendación, métricas de distancia, clustering particional y jerárquico, regresión lineal, árboles de decisión y bosques aleatorios. En cada una se explica lo que se puede realizar y se habla del algoritmo que se utiliza.

2.1. Reglas de recomendación

¿Te has preguntado como Amazon, Netflix u otros servicios te recomiendan productos? Pues la respuesta está en las reglas de recomendación. Las reglas de recomendación se generan a partir de la información de lo que consume un usuario. El algoritmo apriori genera reglas de recomendación utilizando las veces que aparecen los datos que se consumen (frecuencia), el soporte, la confianza y la elevación. El soporte indica qué tan importante es una regla en un conjunto. La confianza indica qué tan fiable es una regla. La elevación indica el aumento de la probabilidad del primer producto al segundo.

En la Figura 2 se muestra la aplicación en la ventana de las reglas de recomendación y lo que se puede realizar. Se tiene una pequeña descripción del componente y una sección para subir datos. Además, se tiene una zona para la gráfica de barras de frecuencia.

En la Figura 3 se observa la parte final de este componente. Se generan reglas de asociación a partir de 3 inputs que representan los parámetros de soporte, elevación y confianza.

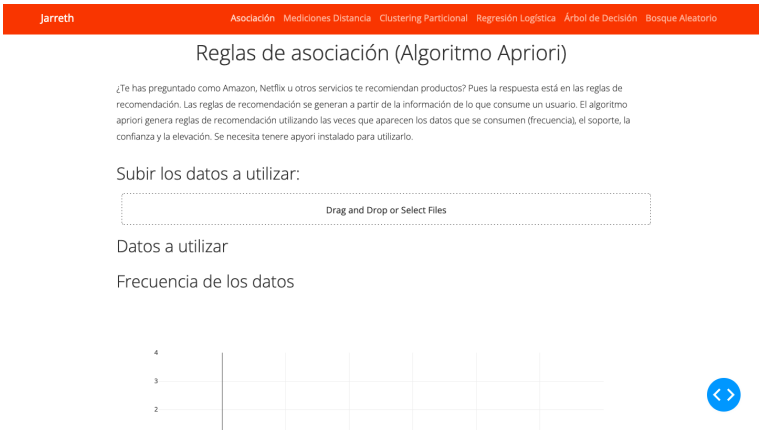

Figura 2: Primera parte Reglas

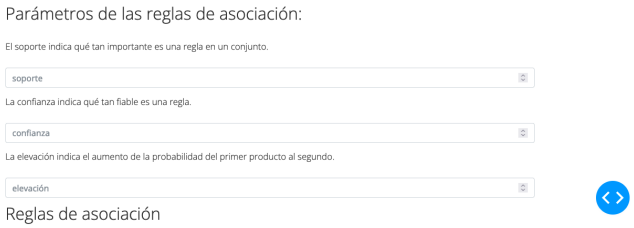

Figura 3: Segunda parte Reglas

2.2. Métricas de distancia

Las métricas de distancia son útiles para medir la distancia entre objetos con diferentes criterios. Si se desea viajar en una ciudad, no se puede solo ir del punto A al punto B en línea recta, ¡existen edificios! Para ello se tienen diferentes métricas para calcular la distancia. La distancia Euclidiana mide la distancia entre dos puntos utilizando una línea recta. La distancia de Chebyshev mide la mayor distancia entre las distintas coordenadas. La distancia Manhattan mide la distancia como una cuadrícula (o una ciudad). La distancia de Minkowski genera una curva entre los dos puntos para medir la distancia.

En la Figura 4 se muestra la primera parte del programa de métricas de distancia. Se tiene una pequeña descripción y la sección de los datos a utilizar. En este apartado se tienen las entradas de dos índices de las filas de los datos que se ingresaron y lo que se seleccionó para generar las distancias.

En la Figura 5 se tienen los cuatro tipos de distancia que se calculan y una pequeña descripción de cada uno de ellos.

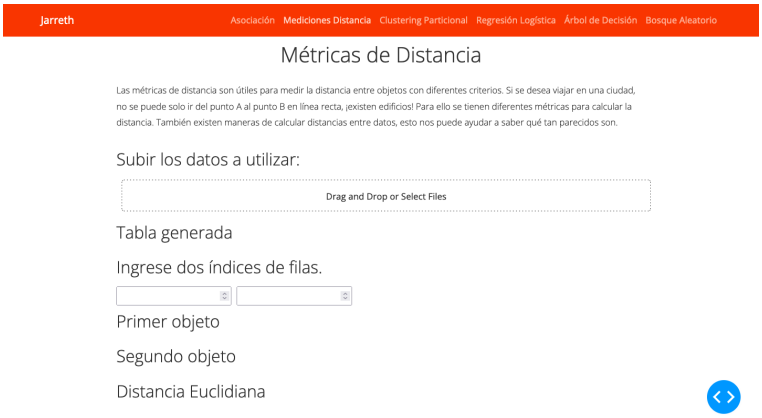

Figura 4: Primera parte distancia

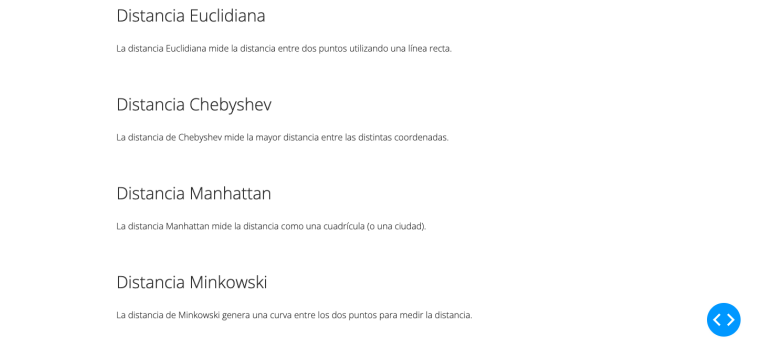

Figura 5: Segunda parte distancia

2.3. Clustering Particional

Algoritmo que genera distintas agrupaciones para clasificar un conjunto de datos. Cada una de las agrupaciones representa un cluster. Estas agrupaciones se generan con ciertos criterios de cercanía (distancias). El problema de este algoritmo es interpretar qué es cada una de las agrupaciones. Se puede elegir la cantidad de grupos en las que se dividen los datos a trabajar.

En la Figura 6 se muestra la primera parte de este componente. Se muestra la sección de descripción y de los datos a utilizar. Además, se tiene un apartado para la matriz de correlación y una pequeña explicación de lo que representa la matriz de correlación.

La segunda parte de este componente se encuentra en la Figura 7. En esta parte se tiene una entrada donde se seleccionan las variables a utilizar en el modelo a partir de los datos que se ingresaron. Además, se puede seleccionar el número de agrupaciones que conforma a este modelo. Por último, se tiene una gráfica donde se mostrarán los resultados utilizando colores.

Figura 6: Primera parte clustering

Figura 7: Segunda parte clustering

2.4. Clustering jerárquico

Este componente se encuentra escondido ya que es muy pesado para su ejecución en la máquina en la que se hizo la aplicación. Se dejó escondida en la aplicación para futuras versiones en las que se pueda revisar cómo se podría optimizar. Para acceder a ella se debe acceder a la ruta /apps/clusterj. No cuenta con descripción y no es parte formal del proyecto. Se recomienda que usuarios principiantes no la utilicen.

2.5. Regresión Logística

La regresión logística permite hacer clasificaciones de datos. Se utiliza una función matemática llamada sigmoide para generar una probabilidad de que el dato sea de una clase o de otra. A partir de esa probabilidad se puede predecir a qué clase pertenece un dato.

En la Figura 8 se muestra la descripción, el apartado de datos a utilizar y la matiz de

correlación del componente de regresión logística.

Figura 8: Primera parte regresión

En la Figura 9 se muestra los valores de entrada para las variables a utilizar para generar el modelo de regresión logística. Además, se tienen botones para generar valores de entrada que corresponden con aquellos de las variables a utilizar. Si se pulsa el segundo botón se genera una predicción de la clase a la que se pertenece. Además, se muestra el *score* del modelo.

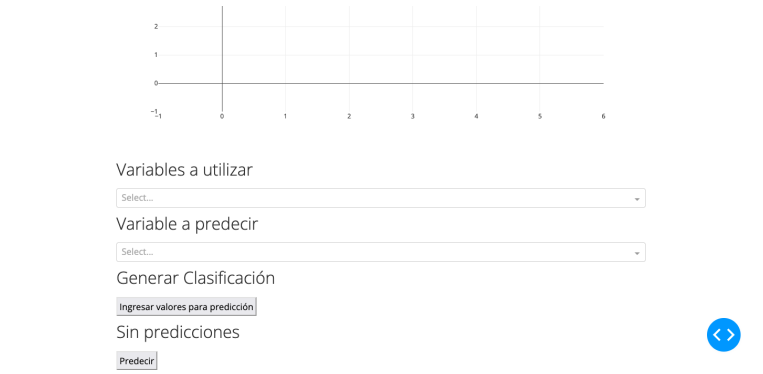

Figura 9: Segunda parte regresión

2.6. Árboles de Decisión

Un árbol de decisión representa un algoritmo que ayuda a tomar decisiones. A partir de un conjunto de decisiones se clasifica a un dato o se predice un valor. Las decisiones agarran una variable del conjunto de datos y preguntan si se es mayor o menor que un valor determinado. Dependiendo de las diferentes decisiones que se tomen se llega a un resultado final. La profundidad máxima de un árbol de decisión representa el máximo de decisiones antes de que se genere un valor final.

En la Figura 10 se muestra la descripción, el apartado de datos a utilizar y la matiz de correlación del componente de árboles de decisión.

Figura 10: Primera parte árbol

En la Figura 11 se tienen entradas para seleccionar las variables a utilizar y las variables a predecir. Además, se tiene una entrada que indica la profundidad máxima de los árboles y una gráfica donde se representará a el modelo por medio de *tree maps*.

Figura 11: Segunda parte árbol

2.7. Bosques Aleatorios

Un bosque aleatorio es la combinación de muchos árboles de decisión que solo operan sobre secciones aleatorias de los datos. Cada uno de los árboles toma una decisión y se genera un promedio o una votación de todos los resultados para obtener el resultado del bosque aleatorio. El número de estimadores representa el número de árboles que se utiliza para generar el bosque. En el programa se selecciona el número de árbol a mostrar de todos los que conforman el bosque.

En la Figura 12 se muestra la descripción, el apartado de datos a utilizar y la matiz de correlación del componente de árboles de decisión.

Figura 12: Primera parte bosques

Para la última parte de este componente se muestra la Figura 13 donde se puede observar las entradas de la variable a predecir y las variables a utilizar. En el componente de bosques aleatorios se reciben dos parámetros: el número de componentes y el número de árbol a mostrar. Por último, se muestra a través de un *tree map* el árbol que se seleccionó.

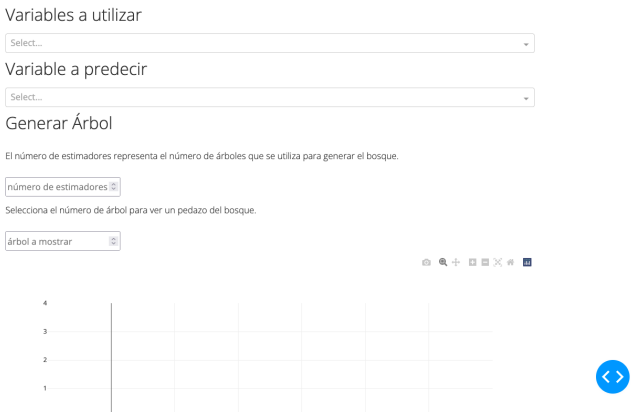

Figura 13: Segunda parte bosques