Proyecto: Minería de datos

Juan Pablo Echeagaray González, Emily Rebeca Méndez Cruz, Grace Aviance Silva Arostegui

***Resumen*—Referencia perrona de este libro [1]**

***Index Terms*—Data Science, Machine Learning, Data Analysis**

I. Introducción

II. Créditos

- Juan Pablo Echeagaray González - Programador / Data Scientist
- Emily Rebeca Méndez Cruz - Programador / Data Scientist
- Grace Aviance Silva Aróstegui - Programador / Data Scientist

III. Modelos de Machine Learning

III-A. Árbol de decisión: Emily Rebeca Méndez Cruz

Los árboles de decisión son un método de aprendizaje supervisado utilizado para la clasificación y la regresión. Este método tiene como objetivo crear un modelo que prediga el valor de una variable de destino mediante el aprendizaje de reglas de decisión simples deducidas de las características de los datos proporcionados. Su estructura puede ser analizada para obtener más información acerca de la relación entre las características y el objeto a predecir.

En esta representación un nodo interno representa un atributo o característica, la rama corresponde una regla de decisión y cada nodo u hoja representa el resultado. El nodo raíz es el nodo superior de un árbol de decisión.

Al momento de crear un árbol, las observaciones de entrenamiento quedan agrupadas en los nodos terminales. Para predecir una nueva observación, se recorre el árbol en función del valor de sus predictores hasta llegar a uno de los nodos terminales.

La metodología de un árbol de decisión es:

- Seleccionar el mejor atributo empleando una medida de selección de características o atributos
- Hacer del atributo seleccionado un nodo de decisión y dividir el conjunto de datos en subconjuntos más pequeños
- Comenzar la selección del árbol repitiendo el proceso en cada atributo hasta que una de estas condiciones se cumpla:
  - Todas las variables pertenecen al mismo valor de atributo
  - Se han acabado los atributos
  - No hay más casos

Juan Pablo Echeagaray González, Emily Rebeca Méndez Cruz, Grace Aviance Silva Arostegui pertencen al Tec de Monterrey campus Monterrey, N.L. C.P. 64849, Mexico

Las reglas de partición ayudan a determinar puntos de ruptura para un conjunto en un nodo dado. La medida de selección de atributos es una heurística y esta proporciona un rango a cada característica, el atributo con la mejor puntuación se selecciona como atributo de división.

Realizar el entrenamiento permite a nuestro modelo tener una base para su futura toma de decisiones, permitiendo obtener un aprendizaje inicial puesto que se le suministra información. Esto proporciona que el modelo pueda plantear correlaciones entre los datos del entrenamiento y los que tiene actualmente guiándolo a la opción correcta.

III-B. Support Vector Machine (SVM) Grace Aviance Silva Arostegui

Support vector machine (SVM) es un algoritmo de aprendizaje supervisado que se utiliza en muchos problemas de clasificación y regresión, e incluso para la detección de valores atípicos [1]. Este modelo de aprendizaje automático se basa en una separación de diferentes clases a través de un hiperplano en un espacio de dimensión superior [3].

Dado un conjunto de muestras (ejemplos de entrenamiento) se etiquetan clases y se entrena una SVM para construir un modelo que prediga la clase de una nueva muestra.

Para la base de datos que tenemos y el análisis que queremos llevar a cabo, en donde el enfoque es considerar en cada uno de los registros de las mujeres cual es el riesgo que tienen para desarrollar cáncer cervical. Dado que en la base de datos el rango de riesgos va de 0 a 4, tenemos así 5 clases de las cuales buscaríamos 5 hiperplanos que tengan el margen lo más ancho posible entre las clases, para así poder entregar lo mejor posible al algoritmo y hacer mejores clasificaciones futuras. Cabe mencionar que utilizamos un 80/20 de los datos para entrenamiento y prueba respectivamente.

El objetivo de este algoritmo es encontrar un hiperplano que separe de la mejor forma posible las clases diferentes de puntos de datos, lo cual implica que el hiperplano tenga un margen lo más amplio posible entre todas las clases, paralela al hiperplano y que no tenga puntos de datos interiores o si acaso permite un número pequeño de clasificaciones erróneas.

III-C. Red Neuronal: Juan Pablo Echeagaray González

Después de inspeccionar el mapa generado hemos notado que hay algunos puntos que parecen tener datos geográficos erróneos, descartar la entrega a estos clientes es algo inaceptable, así que una de las siguientes tareas en el proyecto será desarrollar un método de limpieza efectivo que ayude a mejorar la información geográfica que obtengamos de cada punto.

III-D. Regresión Logística: Juan Pablo Echeagaray González

La regresión logística es otro de los métodos de *machine learning* usados comúnmente para la clasificación de datos. Este algoritmo se usa regularmente para estimar la posibilidad de que una instancia pertenezca a cierta clase; para el caso de clasificación binaria, si dicha probabilidad es mayor al 50 %, se clasifica a esa instancia como perteneciente a la clase positiva. Para generalizar este modelo a problemas de clasificación con n clases ($n > 2$), se calculan n probabilidades de que la instancia pertenezca a la clase i, al final se escoge la que tenga el valor más alto [1] [4].

Al igual que con una regresión lineal, este algoritmo calcula una suma ponderada de los atributos de la instancia, más un término libre; pero lo que el modelo regresa es el resultado de aplicar la función logística a ese número:

$$\hat{p} = \sigma(\mathbf{x^T \Theta}) \tag{1}$$

La función logística tiene un rango de 0 a 1 (por eso su uso como clasificador binario) a su vez se define como:

$$\sigma(t) = \frac{1}{1 + \exp(-t)} \tag{2}$$

Una vez que se cuenta con la estimación de la probabilidad, la salida del modelo queda definida como:

$$\hat{y} = \begin{cases} 0 & \hat{p} < 0{,}5 \\ 1 & \hat{p} \geq 0{,}5 \end{cases} \tag{3}$$

Para entrenar nuestro modelo hemos de encontrar el vector $\mathbf{\Theta} \in \mathbb{R}^{\mathbf{27}}$ (se tienen 27 atributos después de la limpieza de datos) que minimice la función *log loss*, dicha función tiene la siguiente forma:

$$J(\mathbf{\Theta}) = -\frac{\mathbf{1}}{\mathbf{m}} \sum_{\mathbf{i=1}}^{\mathbf{m}} \left[\mathbf{y^{(i)}} \log\left(\mathbf{\hat{p}^{(i)}}\right) + (\mathbf{1} - \mathbf{y^{(i)}}) \log\left(\mathbf{1} - \mathbf{\hat{p}^{(i)}}\right) \right] \tag{4}$$

El índice m de la función 4 representa el número de instancias que tenemos en la base de datos; lo que hace esta función es calcular el promedio del error producido con el vector $\mathbf{\Theta}$. Al final de la fase de entrenamiento se desea haber encontrado el vector que minimice esta función

III-D1. Generalizando a n clases: La regresión logística que clasifica instancias que podrían tener diferentes etiquetas es conocida como *Regresión Logística Múltiple* o *Regresión Softmax* (similar a la función descrita en la sección III-C). Primero se calcula una suma ponderada como en regresión logística para cada una de las posibles clases:

$$s_k(\mathbf{x}) = \mathbf{x^T \Theta^{(k)}} \tag{5}$$

Se usa este valor en la función sigmoide para obtener una probabilidad estimada:

$$\hat{p}_k = \sigma(\mathbf{s(x)})_\mathbf{k} \tag{6}$$

Y al final se escoge la clase con la probabilidad más alta:

$$\hat{y} = \arg\max_k \sigma(s_k(\mathbf{x})) \tag{7}$$

Para nuestro caso de estudio hemos optado por utilizar la librería *scikit-learn* con su regresor logístico documentado en [4]. Los datos de entrenamiento del modelo representarán un 80 % de la base de datos, y la variable que buscaremos predecir será el nivel de riesgo que tiene un individuo dados los 27 atributos que tenemos. Una comparativa de este modelo contra los demás se realizará en la sección IV.

IV. Resultados

V. Conclusiones

V-A. Áreas de mejora

V-B. Modelo seleccionado

VI. Reflexiones

Apéndice A
Datos

Los datos usados en este proyecto pueden descargarse aquí

Apéndice B
Código

El código desarrollado se encuentra en el siguiente repositorio

Apéndice C
Evidencias de trabajo en equipo

Referencias

[1] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Culemborg, Netherlands: Van Duuren Media, 2019.

[2] Sci-kit Learn, "sklearn.tree.DecisionTreeClassifier." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[3] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for machine learning*. Cambridge University Press, 2020.

[4] Sci-kit Learn, "sklearn.linear$_m odel.Logistic Regression.''[Online].Available$: