Primera Entrega Quantum HackMx

Reto propuesto por: NDS Cognitive Labs

Nombre del reto: ¡Mejora, transforma e implementa utilizando tu creatividad y el aprendizaje automático para la detección de fraudes bancarios!

21 de abril del 2021

Equipo: CyberBots
Escuela: ITESM CEM

Integrantes del equipo:
Ana Patricia Islas Mainou
Paulo Ogando Gulias
Cesar Emiliano Palomé Luna
José Luis Madrigal Sánchez

Resumen

En el desarrollo de este avance nos enfocaremos en el algoritmo que se debe seguir para la detección de los fraudes bancarios utilizando machine learning y data science; así mismo, incluiremos algunas especificaciones de la base de datos “creditcard.csv” que hemos descargado de Kaggle para entrenar y probar nuestras neuronas que se encargarán de discriminar si una transacción es o no un fraude bancario.

Introducción

Con el aumento en el número de formas de pago de productos y servicios, también han aumentado los fraudes bancarios y es imperativo que se generen sistemas informáticos y algoritmos capaces de detectar estos fraudes de manera oportuna para evitar una pérdida monetaria y una afectación a los involucrados en dicho acto delictivo.

Por otro lado, durante este siglo, se han desarrollado nuevos métodos para el análisis de grandes cantidades de datos; a estas maneras de analizar y organizar datos se le conoce como Machine Learning y Data Science.

Para la resolución del reto “¡Mejora, transforma e implementa utilizando tu creatividad y el aprendizaje automático para la detección de fraudes bancarios!” propuesto por NDS Cognitive Labs, nosotros implementaremos una red neuronal en Python, entrenada con algoritmos de Machine Learning, que nos permitirá determinar si una transacción es o no un fraude bancario.

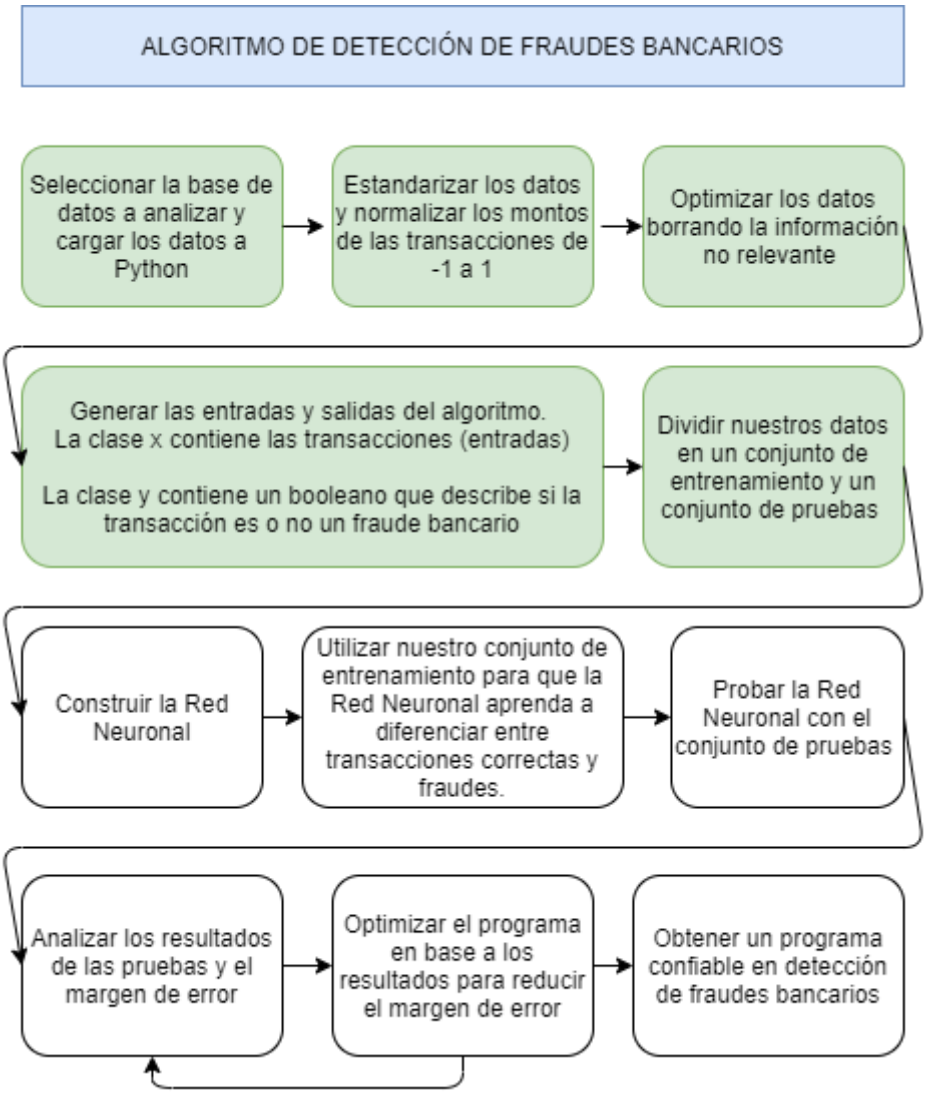
Dentro del lenguaje de programación Python, para nuestra primera entrega, nosotros hacemos uso de las librerías pandas y sklearn. Ya que, estas dos librerías nos permiten cargar y acomodar nuestros datos para poder aplicar el algoritmo de Machine Learning, y subsecuentemente, dividir nuestros datos en un conjunto de entrenamiento y otro de pruebas.

Objetivo

El objetivo de este proyecto es describir el proceso de la implementación de un algoritmo de Machine Learning utilizando redes neuronales y programar dicho algoritmo en Python, utilizando la librería pytorch, para obtener un programa capaz de discriminar entre las transacciones de tarjeta de crédito que son fraudes bancarios y aquellas que no lo son.

Descripción general de la idea para el programa

El siguiente diagrama muestra de manera general los pasos que seguiremos para la creación de nuestro algoritmo de detección de fraudes bancarios. Es importante agregar que los rectángulos en color verde ya están implementados en nuestro código, mientras que los recuadros en blanco aún no se han programado y fungen como guía para la programación.



Cargado de los datos y análisis por PCA

Utilizando la librería “pandas”, se cargó una base de datos que contiene las transacciones bancarias, procesadas por el método de PCA, el cual consiste en seleccionar los datos relevantes para el análisis. Cuando se analizan los datos utilizando el método de PCA “Principal Component Analysis”, se consiguen los componentes principales del conjunto de datos. Estos son la estructura subyacente en los datos y las direcciones donde los datos están más dispersos.

Este análisis considera varios elementos como los eigenvectores y los eigenvalores, los cuales colocan los datos en un nuevo conjunto de dimensiones que nos darán la variación en los datos en ese eigenvector.

Además, se obtiene la covarianza que nos proporciona una relación entre las variables y la media de los datos. Cuando la covarianza entre los datos es positiva implica que los datos son directamente proporcionales, es decir que cuando uno aumenta el otro también lo hace; y cuando la covarianza es negativa implica que la relación entre los datos es inversamente proporcional, es decir que cuando uno aumenta el otro disminuye.

Normalización y optimización de los datos

Utilizando sklearn escalamos los datos del monto de -1 a 1. Cuando se realiza este paso se dice que normalizamos los datos, esto significa que reducimos el rango de los montos para que todos se encuentren entre -1 y 1 manteniendo la relación original que existe entre los montos. De esta manera, evitamos trabajar con números muy grandes o números muy pequeños y optimizamos el algoritmo de Machine Learning.

Se borraron las columnas que no fueran de utilidad para el análisis. Una de las columnas fue el tiempo entre transacciones, ya que esta información sólo aporta el momento en el que se realizaron; y el análisis que buscamos hacer se basa en el número de transacciones. De igual manera se eliminó la columna del monto no normalizado por su amplio rango de 0 a 25,691, el cual nos dejaba un margen de datos muy grande y, al normalizarlo previamente, se optimiza esta parte.

División de los datos en un conjunto de entrenamiento y un conjunto de pruebas

Se genera una matriz “x”, la cual contiene las transacciones y los montos normalizados; y también se genera una matriz “y” que contiene valores booleanos que indican si la transacción es fraude (1) o no es fraude (0).

Para poder implementar el algoritmo de Machine Learning, es necesario dividir nuestras clases mencionadas anteriormente en bases de datos de entrenamiento y bases de datos para pruebas. Utilizando la librería de sklearn.model_selection, destinamos el 70% de los datos para entrenar a nuestra red neuronal y el 30% para probar si el entrenamiento fue exitoso.

Imágen del código programado hasta el momento

La siguiente imágen muestra el código programado hasta el momento, mismo que es explicado a detalle en la sección anterior. Es importante destacar que este código cumple los estándares de programación en Python.

```
15  # Importar Liberiras
16  import pandas as pd
17  from sklearn import preprocessing
18  from sklearn.model_selection import train_test_split
19
20  # Leer la base de datos de transacciones de tarjetas de credito
21  datos = pd.read_csv("creditcard.csv")
22
23  # Standardizzacion de los datos del monto
24  scaler = preprocessing.StandardScaler()
25  # Normalizacion de los montos de -1 a 1
26  datos["NORMALIZADO"] = scaler.fit_transform(datos["Amount"\
27                                                  ].values.reshape(-1, 1))
28
29  # Borar las columnoas de tiempo y monto no normalizado
30  datos = datos.drop(["Amount", "Time"], axis = 1)
31
32  # La clase de y contiene si es fraude (1) o no (0)
33  y = datos["Class"]
34  # La clase de x tiene las transaccioens y el monto normalizado
35  X = datos.drop(["Class"], axis = 1)
36
37  # Crear conjuntos de entrenamiento y de pruebas
38  # El 70 % de los datos es para entrenar la red neuronal
39  # El 30 % de los datos es para probar la red neuronal
40  X_train, X_test, y_train, y_test =train_test_split(X, y, test_size = 0.3, \
41                                                     random_state = 0)
42
```

Bibliografía consultada

Dallas, G. (2021). *Principal Component Analysis 4 Dummies: Eigenvectors, Eigenvalues and Dimension Reduction*. Obtenido de wordpress: https://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-dummies-eigenvectors-eigenvalues-and-dimension-reduction/

Hale, J. (2019). *Scale, Standardize, or Normalize with Scikit-Learn*. Obtenido de Toward Data Science: https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02

Keras.org. (2021). *Deep learning for humans.* Obtenido de keras.io: https://keras.io/

López, J. F. (2021). *Covarianza*. Obtenido de economipedia.com: https://economipedia.com/definiciones/covarianza.html

org, S. L. (2020). *sklearn*. Obtenido de scikitlearn.org: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Pytorch. (2021). *Introducing PyTorch*. Obtenido de pytorch.org: https://pytorch.org/

Sun, L. (2020). *Credit Card Fraud Detection*. Obtenido de towardsdatascience: https://towardsdatascience.com/credit-card-fraud-detection-9bc8db79b956

ULB, M. L. (2018). *Credit Card Fraud Detection*. Obtenido de Kaggle: https://www.kaggle.com/mlg-ulb/creditcardfraud