UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

PIA

Materia: Redes Neuronales

Maestro: José Arturo Berrones Santos

Alejandro Iván Lobo Ortiz

Mat. 1889393

Carrera: ITS

Curso Agosto - Enero 2020
San Nicolás de los Garza , Nuevo León

Índice

Introducción

Para el proyecto integrador de aprendizaje, se desarrollará y se explicará un programa que proponga una solución a un problema de análisis de datos usando técnicas computacionales. Se realizará un pdf con una investigación sobre el tema de aprendizaje o técnica a utilizar, se mencionara las características del problema a estudiar, se explicaran los datos, objetivos, metodologías y resultados. Esta actividad tiene como propósito utilizar los temas vistos en clase aplicados con datos reales.

El aprendizaje de máquina básicamente consiste en algoritmos que facilitan a la computadora aprender. En otras palabras, es una ciencia del área de informática que está vinculada con el tema de inteligencia artificial y como se mencionó antes tiene el objetivo de hacer que los sistemas aprendan. El proceso de aprendizaje no siempre involucra la conciencia, sino técnicas de adquisición de habilidades, agilidad, productividad, entre otras cosas. Muchos de los algoritmos buscan tener el mismo proceso de aprendizaje que un humano.

Existen muchas formas y varias técnicas de aprendizaje, a continuación se platicará sobre los diferentes tipos de algoritmos de aprendizaje máquina, con el propósito de conocer un poco de cada uno.

El aprendizaje supervisado fue uno de los primeros conceptos, se le conoce con ese nombre porque el proceso de aprendizaje de este tipo de algoritmo es a partir de un conjunto de datos previamente etiquetados que son analizados en un de proceso de aprendizaje. Se puede dividir en dos: clasificación, donde la variable de salida es una categoría esperada y regresión donde la variable de salida es un valor real.

Por el otro lado, se encuentra el aprendizaje no supervisado, donde solo tiene datos de entrada y ninguna variable de salida correspondiente, cabe resaltar que los datos no están etiquetados. Una de las formas existentes para resolver problemas no supervisados son: agrupación para encontrar conexiones inherentes en los datos y asocian para descubrir reglas que describen una gran cantidad de datos.

El algoritmo por refuerzo tiene como metodología utilizar constantemente errores estimados como premios o sanciones, es decir se caracteriza por usar

retroalimentación. Esta metodología es practicada con la ayuda de “Q-learning”, también conocido como *quality meaning*, ya que enfatiza la suma importancia de la recompensa al algoritmo para obtener un mejor desempeño al ejecutarse en un sistema. Cada decisión que se toma afecta directamente al entorno y como se mencionó antes el entorno facilita un cierto tipo de comentarios que ayudan al algoritmo a aprender.

Otro algoritmo altamente conocido, es el transducción, este algoritmo es parecido al supervisado pero de una forma más dinámica, ya que para predecir clases el algoritmo hace uso de los ejemplos de entrada y los nuevos ejemplos.

También existe un método donde juntan varios algoritmos para poder encontrar un objetivo de una forma más efectiva, aunque este aparenta ser mejor porque combina varias áreas o algoritmos, puede afectar a ciertos sistemas ya que no siempre es bueno tener más de uno, esto depende de los objetivos del sistema.

Hoy en día, existen muchas muchas áreas donde se puede implementar el aprendizaje máquina, puede ser desde reconocimiento de imagen/productos hasta reconocimiento de voz, predicción de tráfico, carros autónomos, filtros de malware, asistente personal virtual , detección de fraude, gestión de bolsa de valores, diagnóstico médico, traducciones de lenguaje, entre muchos otros más. Sin embargo no se tiene que ir tan lejos para encontrar implementaciones de este tipo de aprendizaje, ya que se puede apreciar en la mayoría de los servicios que se usa hoy en día desde un dispositivos inteligente como en cualquier red social para sacar provecho máximo del usuario y básicamente generar más ingresos

Como Ingeniero en tecnologías de softwares, es importante conocer los diferentes tipos de algoritmos para aprendizaje máquina, ya que son varios y cada uno tiene su respectivo objetivo. Por ende, si se desea tener una buena productividad en la implementación y reducir la cantidad de errores es fundamental comprender las diferencias de los algoritmos. Como se mencionó antes en esta actividad se verá el algoritmo de aprendizaje por refuerzo, desde su definición hasta su historia, ventajas, desventajas, aplicaciones, softwares, entre otras cosas.

Metodología

Definicion

Existen muchas formas para aplicar aprendizaje máquina, para esta ocasión se usará como técnica computacional un tipo de aprendizaje supervisado conocido como clasificación. Segun el doctor Jason Brownlee(2020) este aprendizaje *"se refiere a un problema de modelado predictivo donde se predice una etiqueta de clase para un ejemplo dado de datos de entrada"*. En otras palabras, sirve para poder especificar la clase a la que pertenecen los elementos de datos y se utiliza normalmente cuando la salida tiene valores tanto finitos como discretos.

Hay 2 tipos de clasificación:

- Binomio
- Multi-clase

Tipos de algoritmos

Modelos lineales	Modelos no lineales
Regresión logística Máquinas de vectores de soporte	Vecinos K-más cercanos (KNN) Máquinas de vectores de soporte de kernel Bayes ingenuo Clasificación del árbol de decisión Clasificación aleatoria de bosques

Terminologia

- **Característica:** propiedad individual del fenómeno que se observa.
- **Clasificación binaria:** clasificación con solo dos resultados posibles.
- **Clasificación de clases múltiples:** clasificación con más de dos clases, se asigna a una muestra y solo una etiqueta o objetivo.
- **Entrenar al clasificador:** cada clasificador en sci-kit learn usa el método de ajuste para ajustar el modelo para entrenar.
- **Predecir el objetivo:** para una observación sin etiqueta, el método de predicción devuelve la etiqueta predicha y.
- **Evaluar:** esto básicamente significa la evaluación del modelo

Proceso

Pasos a seguir para crear un modelo.

Paso 1: recopile datos.

Paso 2: preparar los datos.

Paso 3: elige el modelo de clasificación

Paso 4 Entrene el modelo de su máquina..

Paso 5: Evaluación.

Paso 6: Ajuste de parámetros.

Paso 7: predicción o inferencia.

Problema

Definición del problema

El área financiera en cualquier empresa es de suma importancia y no los bancos no son la excepción. Los bancos obtienen ingresos con uso de diferentes estrategias como: préstamos personales, financiamiento del pequeño consumo con tarjetas, descuentos a documentos, adelantos en cuenta corriente, créditos hipotecarios, préstamos prendarios, entre otros. Una de las fuentes de ingresos más importantes para un banco, es cuando un cliente genera una inversión en efectivo para que el banco lo mantenga, esto se le conoce como depósito a plazo. El dinero del cliente se invierte por una tasa de interés durante una cantidad fija de tiempo o plazo. Las ventajas es que el usuario podrá guardar sus recursos en un lugar seguro y podrá obtener un cierto interés positivo al final del periodo, sin embargo el usuario no podrá retirar el dinero durante el periodo.

Los datos están relacionados con campañas de telefónicas de una institución bancaria portuguesa. El objetivo de la clasificación es predecir si el cliente se suscribe a un depósito a plazo (variable y).

Base de datos

La base de datos y el problema a utilizar, se obtuvo de la página kaggle.com. El usuario que publicó la base de datos fue Prakhar Rathi. Para ver más detalles entrar al siguiente link:

https://www.kaggle.com/prakharrathi25/banking-dataset-marketing-targets?select=test.csv

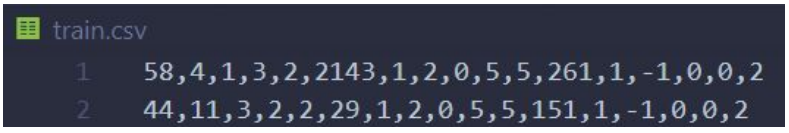
Como se había mencionado antes, los datos están relacionados con las campañas de marketing directo de una institución bancaria portuguesa con llamadas telefónicas. Esto para predecir si el cliente estaría o no suscrito por el cliente. La página brinda los siguientes dos archivos:

- **train.csv:** cuenta con 45,211 filas y 17 columnas ordenadas por fecha
- **test.csv:** 4521 filas y 17 columnas con el 10% de los ejemplos seleccionados al azar de train.csv

La base de datos lucía como la imagen 1.1, sin embargo para facilitar la lectura del programa, se cambiaron las categorías por número, como se muestra en la imagen 1.2. Para conocer los números que representa cada categoría de los atributos, favor de ver la siguiente parte.

```
30,unemployed,married,primary,no,1787,no,no,cellular,19,oct,79,1,-1,0,unknown,no
33,services,married,secondary,no,4789,yes,yes,cellular,11,may,220,1,339,4,failure,no
```

Img 1.1 - *Base de datos antes de actualizarla numéricamente*



Img 1.2 - *Sección de base de datos utilizada*

La base de datos usa los siguientes atributos:

1. **Edad** (numérico)
2. **Trabajo:** (categórico)

1.admin	2.desconocido	3.desempleado	4.gerencia	5.empleada
6.doméstica	7.emprendedor	8.estudiante	9.obrero	
10.autonomo	11.jubilado	12.técnico	13.servicios	

3. **Estado civil (categórico)**

    1.casado

    2.divorciado

    3.soltero

4. **educación (categórica)**

    0.desconocida

    2.secundaria

    1.primaria

    3.terciaria

5. **incumplimiento: ¿tiene crédito en incumplimiento? (binario)** 1.sí, 2.no
6. **saldo: saldo medio anual, en euros (numérico)**
7. **vivienda: ¿tiene préstamo para vivienda?(binario)** 1.sí, 2.no
8. **préstamo: ¿tiene préstamo personal?(binario)** 1.sí, 2.no
9. **contacto: tipo de comunicación de contacto (categórico)**

    1.desconocido, 2.teléfono, 3.celular

10. **día: último día de contacto del mes (numérico)**
11. **mes: último mes de contacto del año (categórico)** 1.jan, . . . , 12.dec
12. **duración: duración del último contacto, en segundos (numérico)**
13. **campaña: número de contactos realizados durante esta campaña y para este cliente (numérico)**
14. **pdays: número de días que pasaron después de que el cliente fue contactado por última vez desde una campaña anterior (numérico, -1 significa que el cliente no fue contactado previamente)**
15. **anterior: número de contactos realizados antes de esta campaña y para este cliente (numérico)**
16. **poutcome: resultado de la campaña de marketing anterior (categórico)** "desconocido", "otro", "fracaso", "éxito")
17. **y - ¿el cliente ha suscrito un depósito a plazo? (binario)** 1.sí, 2.no

Desarrollo

El problema que se presentó entra a la categoría de aprendizaje supervisado, ya que se utiliza un conjunto de datos previamente etiquetados que son analizados en un de proceso de aprendizaje máquina con un método de clasificación. Para realizar una interfaz que pueda generar un modelo, una prueba y crear predicciones, es fundamental la importación de las siguientes librerías. Como se muestra en la siguiente imagen 2.1.

- **Numpy:** en pocas palabras esta librería facilita el uso de funciones matemáticas complejas, además genera números aleatorios, uso de transformadas y rutinas de álgebra lineal.
- **sklearn:** es una librería que ayuda cargar un dataset, predecir datos y crear modelos
- **pickle:** módulo que implementa protocolos binarios.
- **time:** módulo que facilita funciones relacionadas al tiempo.
- **tkinter:** es una de las bibliotecas de interfaz gráfica de Python más populares para el desarrollo de aplicaciones de escritorio. Es una combinación del framework GUI estándar de TK y Python.

```
import tkinter as tk
from tkinter import Entry, LabelFrame
from tkinter import messagebox
from tkinter import ttk
from tkinter.constants import COMMAND, END, VERTICAL
#-------------------------generación modelo, predicción
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
import pickle
import time
```

Img 2.1 - *Librerías*

Al ejecutar el programa, primero se corren estas líneas donde se va ejecutar primero la función de MyApp() para crear la ventana donde se va realizar la ventana. Luego se ponen los atributos como tamaño y título. Como se muestra en la imagen 2.2.

```
app = MyApp()
app.geometry("720x600")
app.title("    BANKPREDICTOR")
app.resizable(False,False)
app.mainloop()
```

Img 2.2 - *MyApp*

La función MyApp tiene como objetivo configurar parámetros iniciales y el menú para las interfaces del programa. Con sus respectivos comandos, para que sean funcionales. Como se muestra en la imagen 2.3 y 2.4.

```
def __init__(self, *args, **kwargs, ):
        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)
        container.pack(side="top", fill="both", expand=True)
        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}
        # -------------------------- menu
        menu = tk.Menu(container)
        betting = tk.Menu(menu, tearoff=0)
        menu.add_cascade(menu=betting, label="Opciones")
        betting.add_command(label="Modelo Test",command=lambda: self.show_frame(Startpage))
        betting.add_command(label="Predicción",command=lambda: self.show_frame(PagePrediction))

        tk.Tk.config(self, menu=menu)

        for F in (Startpage, PagePrediction):
            frame = F(container, self)
            self.frames[F] = frame
            frame.grid(row=0, column=0, sticky="nsew")
            frame.config(bg="#3CB371")

        self.show_frame(Startpage)
```

Img 2.3 - *Myapp código*

```python
def show_frame(self, cont):
        frame = self.frames[cont]
        frame.tkraise()
```

Img 2.4 - *función para las interfaces*

Dentro de la primera clase (Startpage)para la primera interfaz, se harán uso de labels, campos de entrada y botones, con el fin de que sea más atractiva para el usuario. El modelo va buscar el menor error dentro un rango de diferentes redes neuronales, el entry de la interfaz representa el máximo de neuronas a analizar. El mínimo de neuronas en la capa oculta será de 5 y si el usuario solo quiere 5, necesitará escribir 6, de lo contrario si quiere analizar diferentes cantidades para encontrar el mínimo error deberá ingresar una cantidad mayor al mínimo. Como se muestra en la imagen 2.5. Además están los dos botones, generador de modelos y el de pruebas, con su respectivo comando que se analizará en los siguientes párrafos.

```python
def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        tt = tk.Label(self, text="BIENVENIDO A BANKPREDICTOR",bg="#3CB371",fg='white',font=('Helvetica', 18, 'bold')).place(x=170
,y=35)
        div = tk.Label(self, text="_____________________________________________________",bg="#3CB371",fg='black',font=('
Helvetica', 15, 'bold')).place(x=50,y=110)
        l1 = tk.Label(self, text="BANKPREDICTOR busca predecir un el cliente se suscribirá \n\n
a un depósito a plazo (term deposit).",bg="#3CB371",fg='black')
        l1.place(x=190,y=110)
        div2 = tk.Label(self, text="_____________________________________________________",bg="#3CB371",fg='black',font=('
Helvetica', 15, 'bold')).place(x=50,y=400)

        t1 = tk.Label(self, text="OBJETIVOS",bg="#3CB371",fg='black',font=('Helvetica', 12, 'bold')).place(x=312,y=190)
        t2 = tk.Label(self, text=" Neuronas (x)               ",bg="#3CB371",fg='black',font=('Helvetica', 8, 'italic')).place(x
=320,y=280)
        t3 = tk.Label(self, text="Se va generar un modelo con el menor error posible del rango 5 a 'x'. \n
Insertar en el Entry la cantidad maxima de neuroas en el hidden layer a analizar",bg="#3CB371",fg='black').place(x=170,y=240)
        #-------------------------- Entry
        self.e_nh1 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
        self.e_nh1.place(x=320,y=300)

        # -------------------------- generar modelo
        act = tk.Button(self, text="Generar Modelo",height = 1, width = 14,bg="#fedc56",command= self.gModelo)
        act.place(x=250,y=330)
        # -------------------------- test
        act = tk.Button(self, text="Crear Test",height = 1, width = 14,bg="#fedc56",command= self.gTest)
        act.place(x=370,y=330)

        #-----------------------------------aviso legal
        w = tk.Canvas(self, width=610, bg="#3CB371",height=100).place(x=50,y=460)
        e = tk.Label(self, text="AVISO LEGAL",bg="#3CB371",fg='black',font=('Helvetica', 12, 'bold')).place(x=300,y=465)
        sub1 = tk.Label(self, text="
 Bankpredictor excluye cualquier responsabilidad por los daños y perjuicios de toda naturaleza que pudieran deberse\n
 a la mala utilización del Servicio. Por motivos de seguridad, Bankprediction no recopila\n
 ni almacena información confidencial de su empresa",bg="#3CB371",fg='black',font=('Helvetica', 8, 'italic')).place(x=60,y=500)
```

Img 2.5 - *Start page*

Al dar clic al botón generar modelo, primero se obtiene el string del único Entry de la interfaz, después se checa que ese mismo valor no este vacio, de ser así se le pedirá al usuario que ingrese un valor, de lo contrario se continúa. Primero se retroalimenta al usuario con un diálogo que se está generando el modelo. Se genera el modelo y se imprimen los errores, la matriz de confusión y el reporte de clasificación. Para generar el modelo, se usaron los siguientes pasos:

1. Primero se lee la data del archivo .csv “train.csv”, hay que tomar en cuenta que tanto la “x” como la “y” se encuentra en el mismo archivo, con ayuda de la función .loadtxt()
2. Se construyen conjuntos de entrenamiento y prueba al azar, con la función train_test_split() con sus respectivos parámetros. Hay se define que solo el 30 por ciento de los datos se usarán para el test.
3. Se inicializa un tuple para guardar los errores de cada cantidad diferente neuronas dentro del rango.
4. Obteniendo la variable del Entry de la interfaz que define la cantidad máxima del rango de neuronas, se usa esa variable en un for para poder analizar todos los modelos con diferentes cantidades de neutrones en la capa oculta para encontrar el mínimo.
    a. Dentro del for …
    b. Se crea el modelo de clasificación con la cantidad correspondiente de neuronas en la capa oculta.
    c. Se usa el modelo entrenado para predecir las salidas sobre el conjunto de prueba.
    d. Se analizará el error con el accuracy score y se guardará en un tuple empaquetado de listas donde almacenará la cantidad de neuronas del hidden layer que se utilizó y su respectivo error.
5. Después de acabar el for, se agarra con la función min() el que tiene menor error dentro del tuple.
6. En un cuadro de diálogo, se imprime el que tiene el menor error y se imprime en la consola el reporte de clasificación y la matriz de confusión.
7. Se guarda el modelo en un archivo llamado “trained_modelBankMLP.sav”

```
def gModelo(self):
    ehl = self.e_nhl.get()
    if (ehl == ""):
        messagebox.showinfo("Estatus", "
Se requiere insertar la cantidad máxima de neuronas (x), porque se va buscar el menor error de 5 a x neuronas en hidden layer")
    else:
        messagebox.showinfo("Estatus", "Generando el modelo con la menor catidad de error del rango de neuronas de hl.")
        # Leyendo el csv sacando los datos de X y Y
        x = np.loadtxt('train.csv', delimiter=',', usecols=range(16))
        y = np.loadtxt('train.csv', delimiter=',', usecols=(16, ))

        # se construyen conjuntos de entrenamiento y prueba al azar
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.30, random_state=3)
        eTupple =[]
        # hlmax = int(input('Se buscará el menor error desde el hidden Layer 5 hasta: '))
        for hl in range(5,int(ehl)):
            # se usa el conjunto de prueba para entrenar el clasificador
            clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(hl,), random_state=1)
            clf.fit(x_train, y_train)

            # se usa el modelo entrenado para predecir las salidas sobre el conjunto de prueba
            predicted = clf.predict(x_test)

            # se calcula el error sobre el conjunto de prueba
            error = 1 - accuracy_score(y_test, predicted)
            eTupple.append((error,hl))

        hl = min(eTupple)
        # print(eTupple)
        clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(hl[1],), random_state=1)
        clf.fit(x_train, y_train)

        #Resultados
        predicted = clf.predict(x_test)
        print('\n--------BANK-------\nErrores de diferentes cantidades de neuronas en hl (error,#hl): {}\n'.format((eTupple)))
        messagebox.showinfo("Estatus", 'Error menor sobre el conjunto de prueba: {}, HL: {}'.format(hl[0],hl[1]))
        messagebox.showinfo("Estatus", 'En la consola se mostrará el reporte de clasificación y la matrix de confusión')
        print('Reporte de clasificación: \n',metrics.classification_report(y_test, predicted))
        print('Matrix de confusión: \n',metrics.confusion_matrix(y_test, predicted))

        # se guarda el modelo entrenado para uso posterior
        filename = 'trained_modelBankMLP.sav'
        pickle.dump(clf, open(filename, 'wb'))
        self.e_nhl.delete(0,'end')
```

Img 2.6 - *Start page (gModelo)*

Al presionar el botón de “Crear test”, primero notifica al usuario que se está creando una prueba de error en data set con un diálogo. Después se leen el archivo “test.csv”, así como el de train la “x” y “y” están dentro de un mismo archivo. Luego se carga el modelo anteriormente creado con el nombre de “trained_modelBankMLP.sav”. Se usa el modelo para predecir las salidas y se saca el error restando 1 - el accuracy_score. Con la finalidad de mostrar al usuario el error de la prueba en el dataset dentro de un diálogo. Como se muestra en la imagen 2.7.

```
def gTest(self):
    messagebox.showinfo("Estatus", "
Sacando el error del archivo Test con 4521 ejemplos de datos con el modelo")
    x = np.loadtxt('test.csv', delimiter=',', usecols=range(16))
    y = np.loadtxt('test.csv', delimiter=',', usecols=(16, ))

    # se carga la red neuronal entrenada
    clf = pickle.load(open('trained_modelBankMLP.sav', 'rb'))

    # se usa el modelo entrenado para predecir las salidas
    predicted = clf.predict(x)

    # se calcula el error de prediccion
    error = 1 - accuracy_score(y, predicted)
    messagebox.showinfo("Estatus", 'Error de prediccion del test: {}'.format(error))
```

Img 2.7 - *Start page (gTest)*

En la siguiente clase (Page Prediction) para la página de predicción, es una interfaz que cuenta con una serie de labels, Entrys y etiquetas para que sea más digerible el sistema para los usuarios. Como se muestra en la imagen 2.8.

```
class PagePrediction(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        ll = tk.Label(self, text="PREDICCIÓN",bg="#3CB371",fg='white',font=('Helvetica', 25, 'bold')).place(x=
250,y=35)

        div = tk.Label(self, text="_____",bg="#3CB371",fg='
black',font=('Helvetica', 15, 'bold')).place(x=50,y=110)
        div2 = tk.Label(self, text="_____",bg="#3CB371",fg='
black',font=('Helvetica', 15, 'bold')).place(x=50,y=400)


        sp_op = tk.Label(self, text="Edad",bg="#3CB371",fg='black').place(x=70,y=170)
        sp_op = tk.Label(self, text="Trabajo",bg="#3CB371",fg='black').place(x=70,y=200)
        sp_op = tk.Label(self, text="Estado Civil",bg="#3CB371",fg='black').place(x=70,y=230)
        sp_op = tk.Label(self, text="Educación",bg="#3CB371",fg='black').place(x=70,y=260)

        sp_op = tk.Label(self, text="¿Cuenta con tarjeta de credito?",bg="#3CB371",fg='black').place(x=70,y=290)
        sp_op = tk.Label(self, text="Salario promedio anual",bg="#3CB371",fg='black').place(x=70,y=320)
        sp_op = tk.Label(self, text="¿Tiene préstamo para vivienda?",bg="#3CB371",fg='black').place(x=70,y=350)
        sp_op = tk.Label(self, text="¿Tiene préstamo personal?",bg="#3CB371",fg='black').place(x=70,y=380)
```

Img 2.8 - *PagePrediction*

Después de llenar todos los campos, se da clic al botón. Lo primero que hace, es obtener todos los campos para juntarlos en tuple para poder hacer la predicción después. Luego se asegura el sistema que los campos no estén vacíos. Para hacer la predicción se usa el mismo método que se usó para sacar el error en la prueba, para ello se carga el dataset de prueba y el modelo entrenado, con el fin de predecir las salidas. Una vez teniendo las salidas, el resultado se imprime en el campo de predicción para que el usuario lo pueda apreciar. Como se muestra en la imagen 2.9 y 2.10. Además, si se presiona el botón de “vaciar campos”, se vacían los campos, como se muestra en la imagen 2.11.

```
def pprediction(self):
       p1 = self.p1.get()
       p2 = self.p2.get()
       p3 = self.p3.get()
       p4 = self.p4.get()
       p5 = self.p5.get()
       p6 = self.p6.get()
       p7 = self.p7.get()
       p8 = self.p8.get()

       p9 = self.p9.get()
       p10 = self.p10.get()
       p11 = self.p11.get()
       p12 = self.p12.get()
       p13 = self.p13.get()
       p14 = self.p14.get()
       p15 = self.p15.get()
       p16 = self.p16.get()
       if (p1 == "" or p2 == "" or p3 == "" or p4 == "" or p5 == "" or p6 == ""
or p7 == "" or p8 == "" or p9 == "" or p10 == "" or p11 == "" or p12 == "" or
 p13 == "" or p14 == "" or p15 == "" or p16 == ""):
           messagebox.showinfo("Estatus", "
Se requiere llenar todos los campos para hacer una predicción")
       else:
           pred = []
           pred.append(int(p1))
           pred.append(int(p2))
           pred.append(int(p3))
           pred.append(int(p4))
           pred.append(int(p5))
           pred.append(int(p6))
           pred.append(int(p7))
           pred.append(int(p8))
           pred.append(int(p9))
           pred.append(int(p10))
           pred.append(int(p11))
           pred.append(int(p12))
           pred.append(int(p13))
           pred.append(int(p14))
           pred.append(int(p15))
           pred.append(int(p16))

           x = np.loadtxt('test.csv', delimiter=',', usecols=range(16))
           y = np.loadtxt('test.csv', delimiter=',', usecols=(16, ))

           # se carga la red neuronal entrenada
           clf = pickle.load(open('trained_modelBankMLP.sav', 'rb'))

           # se usa el modelo entrenado para predecir las salidas
           predicted = clf.predict(x)

           #------------------predicción
           Xnew = [pred]
           ynew = clf.predict(Xnew)
           ynew=ynew[0]
           print("predicción: ",(ynew))
           if ynew == 2:
               self.pp.config(state=tk.NORMAL)
               self.pp.insert(0, "NO")
           else:
               self.pp.config(state=tk.NORMAL)
               self.pp.insert(0, "YES")
```

Img 2.9 - *PagePrediction*

```
self.p1 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p1.place(x=240,y=170)
self.p2 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p2.place(x=240,y=200)
self.p3 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p3.place(x=240,y=230)
self.p4 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p4.place(x=240,y=260)

self.p5 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p5.place(x=240,y=290)
self.p6 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p6.place(x=240,y=320)
self.p7 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p7.place(x=240,y=350)
self.p8 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p8.place(x=240,y=380)

sp_op = tk.Label(self, text="Tipo de comunicación",bg="#3CB371",fg='black').place(x=380,y=170)
sp_op = tk.Label(self, text="Último dia de comunicación",bg="#3CB371",fg='black').place(x=380,y=200)
sp_op = tk.Label(self, text="Último mes de comunicación",bg="#3CB371",fg='black').place(x=380,y=230)
sp_op = tk.Label(self, text="Duración",bg="#3CB371",fg='black').place(x=380,y=260)

sp_op = tk.Label(self, text="Cantidad de campaña",bg="#3CB371",fg='black').place(x=380,y=290)
sp_op = tk.Label(self, text="Días desde la última com.",bg="#3CB371",fg='black').place(x=380,y=320)
sp_op = tk.Label(self, text="Cantidad de comunicación\n antes de la campaña",bg="#3CB371",fg='black').place(x=380,y=350)
sp_op = tk.Label(self, text="Resultado Marketing anterior",bg="#3CB371",fg='black').place(x=380,y=380)

self.p9 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p9.place(x=555,y=170)
self.p10 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p10.place(x=555,y=200)
self.p11 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p11.place(x=555,y=230)
self.p12 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p12.place(x=555,y=260)

self.p13 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p13.place(x=555,y=290)
self.p14 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p14.place(x=555,y=320)
self.p15 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p15.place(x=555,y=350)
self.p16 = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.p16.place(x=555,y=380)

e_pp = tk.Label(self, text="Predicción: ",bg="#3CB371",fg='black').place(x=70,y=440)

self.pp = tk.Entry(self,bg="#d9dcd6",fg="#16425B", width=12)
self.pp.place(x=240,y=440)

self.btn_p = tk.Button(self, text="Crear predicción",bg="#fedc56",
                command=lambda: [self.pprediction()])
self.btn_p.place(x=534,y=440)

self.btn_p = tk.Button(self, text="Vaciar campos",bg="#CD5C5C",
                command=lambda: [self.vc()])
self.btn_p.place(x=470,y=440)
```

Img 2.10 - *PagePrediction*

```
def vc(self):
    self.p1.delete(0,'end')
    self.p2.delete(0,'end')
    self.p3.delete(0,'end')
    self.p4.delete(0,'end')
    self.p5.delete(0,'end')
    self.p6.delete(0,'end')
    self.p7.delete(0,'end')
    self.p8.delete(0,'end')
    self.p9.delete(0,'end')
    self.p10.delete(0,'end')
    self.p11.delete(0,'end')
    self.p12.delete(0,'end')
    self.p13.delete(0,'end')
    self.p14.delete(0,'end')
    self.p15.delete(0,'end')
    self.p16.delete(0,'end')
    self.p16.delete(0,'end')
    self.pp.delete(0,'end')
    pass
```

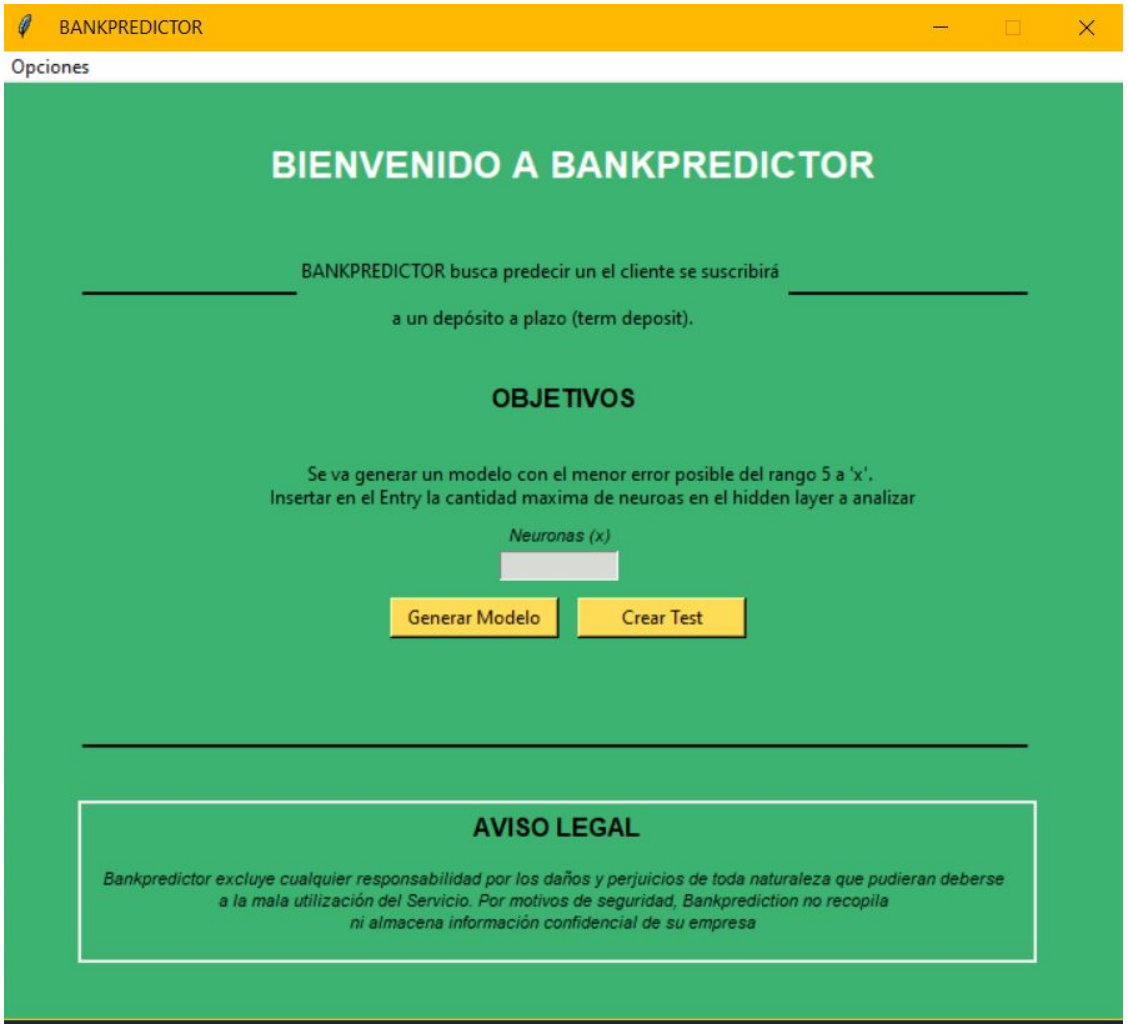
Img 2.11 - *Vaciar campos*

Ejecución y resultados

Arriba en la esquina superior izquierda se puede apreciar un menú de dos opciones, modelo test y predicción. Como se muestra en la imagen 3.1.

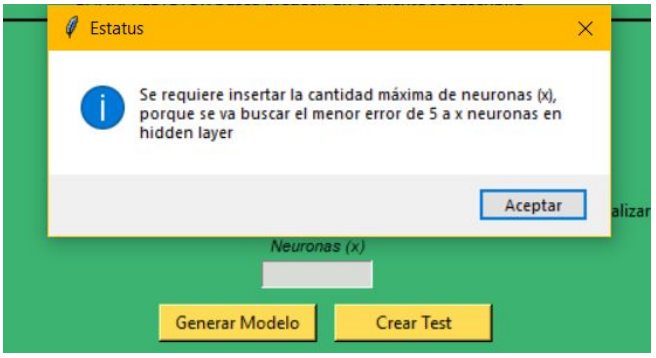
Img 3.1 - *Menú*

La primera interfaz que el usuario verá o si se da clic en la opción “Modelo Test” del menú se verá la interfaz de la imagen 3.2. Cuenta con un mensaje de bienvenida, un slogan, instrucciones para llenar el campo, dos botones y un aviso legal.



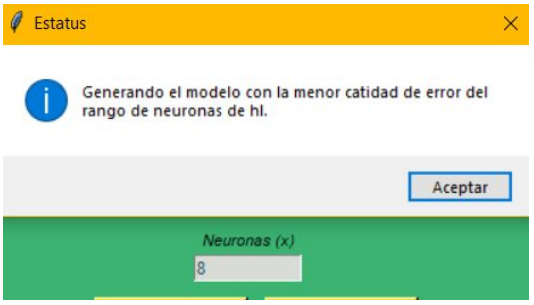
Img 3.2 - *Interfaz 1*

Si el usuario genera el modelo sin insertar el valor máximo del rango para las neuronas, saldrá un mensaje como en la imagen 3.3.



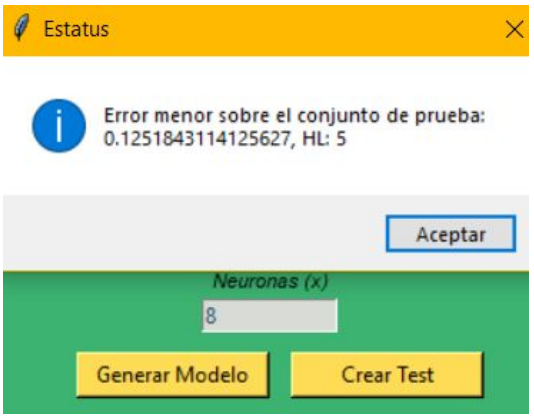
Img 3.3 - *Retroalimentación*

De lo contrario, si se llena el campo y se da clic en el botón de generar, el sistema retroalimentar al usuario que se está generando el modelo como en la siguiente mensaje de la imagen 3.4



Img 3.4 - *Retroalimentación 2*

Al terminar el modelo, se mostrará el error menor sobre el conjunto de prueba dentro del rango que el usuario escogió. En este caso se analizaron los modelos entre el cinco y el ocho neuronas en la capa oculta, después del proceso de selección de error mínimo se dice que el modelo con menor error, es el que tiene 5 neuronas en la capa oculta con un error del 15%. Esto habla de un buen modelo entrenado, como se muestra en la imagen 3.5.



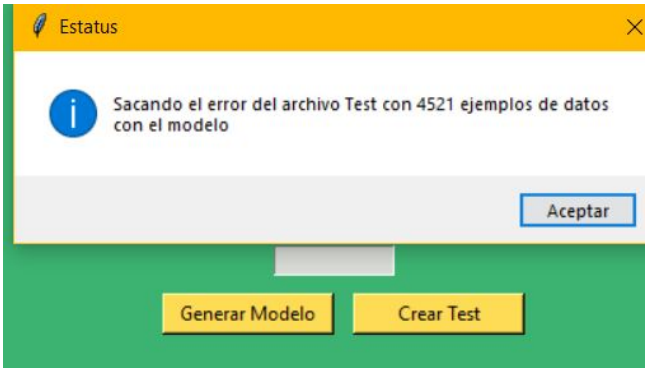
Img 3.5 - *Error*

En la imagen 3.6 se puede apreciar un arreglo de diferentes modelos con su error y la cantidad de diferentes neuronas del rango que el usuario escogió, para encontrar el menor error posible. Como se puede observar, efectivamente el modelo con 5 neuronas en la capa oculta es el que tiene menos porcentaje de error. Además, se imprimió el reporte de clasificación donde se ve que es menos probable que el usuario tenga un depósito a plazo (term deposit).

```
--------BANK-------
Errores de diferentes cantidades de neuronas en hl (error,#hl): [(0.1251843114125627, 5), (0.13852845768209965, 6), (0.27624594514892364, 7)]

Reporte de clasificación:
              precision    recall  f1-score   support

         1.0       0.25      0.03      0.06      1595
         2.0       0.88      0.99      0.93     11969

    accuracy                           0.87     13564
   macro avg       0.57      0.51      0.50     13564
weighted avg       0.81      0.87      0.83     13564

Matrix de confusión:
 [[   53  1542]
 [  156 11813]]
```

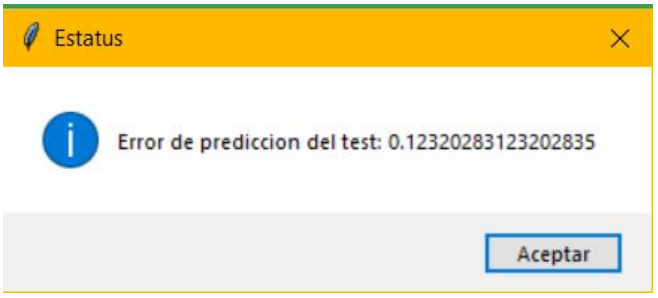
Img 3.6 - *Resultados del modelo entrenado*

Al dar clic “crear test”, retroalimenta al usuario de forma inmediata que se está realizando la prueba para observar el error del dataset de prueba con el modelo entrenado.



Img 3.7 - *Retroalimentación 3*

Al dar clic aceptar, mostrará otro diálogo con el error de la prueba. En este caso, al analizar el dataset con el modelo entrenado con 5 capas se puede apreciar que cuenta con un porcentaje de error del 12%, que vendría siendo muy bueno.



Img 3.8 - *Error de la prueba*

Se llenan todos los campos, que vendrían siendo todos los atributos del dataset. Al llenarse, se da clic en crear predicción y despliega el resultado en el campo de abajo. Como se había mencionado en los resultados de la imagen 3.6, es muy probable que salga “no”, es decir que el cliente no quiere tener depósito a plazo. Además, como se puede apreciar en los errores del modelo y la prueba, el modelo cuenta con un porcentaje bajo de error, por lo tanto es muy probable que sea cierto.

Img 3.9 - *Predicción*

Referencias

- Videos, artículos y programas que se proporcionaron para este tema en el curso Agosto - Diciembre 2020
- Python (und) pickle. Recuperado el 10/12 /2020 de https://docs.python.org/3/library/pickle.html
- scikit Learn (und) sklearn. Recuperado el 10/12 /2020 de https://scikit-learn.org/stable/
- Python (und) time. Recuperado el 10/12 /2020 de https://docs.python.org/3/library/pickle.html
- Numpy (und) Numpy. Recuperado el 10/12/2020 de https://numpy.org/
- S. Moro, P. Cortez and P. Rita. A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, Elsevier, 62:22-31, June 2014
- S. Moro, R. Laureano and P. Cortez. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology.
- In P. Novais et al. (Eds.), Proceedings of the European Simulation and Modelling Conference - ESM'2011, pp. 117-121, Guimarães, Portugal, October, 2011. EUROSIS.
- iProfesional (2008) ¿Cúal es la principal fuente de ingresos de los bancos?. . Recuperado el 10/12 /2020 de https://www.iprofesional.com/finanzas/61367-cual-es-la-principal-fuente-de-ingresos-de-los-bancos
- simplilearn (und) Classification machine learning tutorial. Recuperado el 11/12/2020 de https://www.simplilearn.com/classification-machine-learning-tutorial
- Jason, B (und) Classification. Recuperado el 11/12/2020 de https://machinelearningmastery.com/types-of-classification-in-machine-learning/#:~:text=In%20machine%20learning%2C%20classification%20refers,one%20of%20the%20known%20characters.
- AI (2020) The 7 Key Steps To Build Your Machine Learning Model. Recuperado el 11/12/2020 de https://analyticsindiamag.com/the-7-key-steps-to-build-your-machine-learning-model/