UNIVERSIDAD POLITECNICA SALESIANA

SEDE CUENCA

CARRERA: *INGENIERIA DE SISTEMAS*

Nombre: Bryam Gabriel Mora Lituma

Materia: Sistemas Expertos

Fecha:18/01/2021

Caso práctico de implementación de un sistema de razonamiento basadoven casos.

► Como caso práctico se implementará un sistema CBR básico para determinar la calidad del vino rojo.

► Para ello, se trabajará con el corpus Wine Quality Data Set.

► El corpus se compone de un total de 1599 muestras de vino rojo que contienen información de pruebas fisicoquímicas realizadas en vinos rojos.

► La información detallada del corpus y las técnicas que aplicaron los autores se puede encontrar en el siguiente enlace:

https://www.sciencedirect.com/science/article/abs/pii/S0167923609001377?via%3Dihub

Para ello, se deberá considerar lo siguiente:

Se tienen los siguientes atributos del vino:

1 - fixed acidity
2 - volatile acidity
3 - citric acid
4 - residual sugar
5 - chlorides
6 - free sulfur dioxide
7 - total sulfur dioxide
8 - density
9 - pH
10 - sulphates
11 - alcohol
Variable de salida:
12 - quality (puntaje entre 0 y 10)

Las tareas a realizar son las siguientes:

► Preprocesar los datos del corpus de acuerdo a las sugerencias desarrolladas por wguillen.

https://github.com/wguilen/red-wine-quality-cbr/tree/master/presentation

► Aplicar la técnicas de los vecinos más cercanos indicada en clase y empleando la fórmula propuesta por wguillen.

$$\text{Similaridade}(A1C1, A1C2) = 1 - \frac{|A1C2 - A1C1|}{(\text{val max} - \text{val min})}$$

Intervalo de variação!

► Desarrollar una pequeña interfaz en Python u otro lenguaje donde se coloquen los atributos y el sistema indique la calidad del vino.

► Realizar un pequeño informe del trabajo desarrollado, considerando los aspectos principales y qué tan preciso es el sistema.

DESARROLLO

El sistema sera desarrollado en el lenguaje de Python, se utilizara varias librerias propias de este lenguaje que nos ayudaran a realizar el programa. Procederemos a ver el codigo fuente y de nuestro programa en los siguientes pasos.

Librerias Utilizadas.

Tkinter: Proporciona un conjunto de herramientas robusto e independiente de la plataforma para administrar ventanas.

Pandas: Es una biblioteca de software escrita para el lenguaje de programación Python para la manipulación y el análisis de datos. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series de tiempo.

Operator: Es un módulo que exporta un conjunto de funciones eficientes correspondientes a los operadores intrínsecos de Python. Por ejemplo operator.add(x, y), es equivalente a la expresión x+y.

1. Primero importamos las librerias

```
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
import pandas as pd
import operator
```

2. Creamos la interfaz de nuestra aplicacion de la siguiente manera.

```
raiz.geometry('600x270')  # anchura x altura

raiz.title('Calidad de vinos')

Label(raiz, text="Analisis de calidad de vino").place(x=200, y=0)

Label(raiz, text="Fixed Acidy").place(x=0, y=25)
tfa = Spinbox(raiz, from_=0.0, to=15.9, width=5, increment=0.1)
tfa.place(x=150, y=25)
ctfa = ttk.Combobox(raiz, values=lista, width=5)
ctfa.place(x=225, y=25)
ctfa.current(3)

Label(raiz, text="Volatily Acidy").place(x=308, y=25)
tva = Spinbox(raiz, from_=0.0, to=1.58, width=5, increment=0.01)
tva.place(x=450, y=25)
ctva = ttk.Combobox(raiz, values=lista, width=5)
ctva.place(x=525, y=25)
ctva.current(3)

Label(raiz, text="Citric Acid").place(x=0, y=50)
tca = Spinbox(raiz, from_=0.0, to=1.0, width=5, increment=0.1)
tca.place(x=150, y=50)
ctca = ttk.Combobox(raiz, values=lista, width=5)
ctca.place(x=225, y=50)
ctca.current(3)

Label(raiz, text="Residual Sugar").place(x=308, y=50)
trs = Spinbox(raiz, from_=0.0, to=13.9, width=5, increment=0.1)
trs.place(x=450, y=50)
ctrs = ttk.Combobox(raiz, values=lista, width=5)
ctrs.place(x=525, y=50)
ctrs.current(5)

Label(raiz, text="Chlorides").place(x=0, y=75)
```

```
tc = Spinbox(raiz, from_=0.0, to=0.611, width=5, increment=0.001)
tc.place(x=150, y=75)
ctc = ttk.Combobox(raiz, values=lista, width=5)
ctc.place(x=225, y=75)
ctc.current(1)

Label(raiz, text="Free Sulfur Dioxide").place(x=308, y=75)
tfsd = Spinbox(raiz, from_=0.0, to=72.0, width=5, increment=1.0)
tfsd.place(x=450, y=75)
ctfsd = ttk.Combobox(raiz, values=lista, width=5)
ctfsd.place(x=525, y=75)
ctfsd.current(1)

Label(raiz, text="Total Sulfure Dioxide").place(x=0, y=100)
ttsd = Spinbox(raiz, from_=0.0, to=289.0, width=5, increment=1)
ttsd.place(x=150, y=100)
cttsd = ttk.Combobox(raiz, values=lista, width=5)
cttsd.place(x=225, y=100)
cttsd.current(1)

Label(raiz, text="Density").place(x=308, y=100)
td = Spinbox(raiz, from_=0.0, to=1.0000, width=6, increment=0.0001)
td.place(x=450, y=100)
ctd = ttk.Combobox(raiz, values=lista, width=5)
ctd.place(x=525, y=100)
ctd.current(1)

Label(raiz, text="pH").place(x=0, y=125)
tph = Spinbox(raiz, from_=0.0, to=4.01, width=5, increment=0.01)
tph.place(x=150, y=125)
ctph = ttk.Combobox(raiz, values=lista, width=5)
ctph.place(x=225, y=125)
ctph.current(6)

Label(raiz, text="Sulphates").place(x=308, y=125)
```

```
ts = Spinbox(raiz, from_=0.0, to=2.0, width=5, increment=0.01)
ts.place(x=450, y=125)
cts = ttk.Combobox(raiz, values=lista, width=5)
cts.place(x=525, y=125)
cts.current(1)

Label(raiz, text="Alcohol").place(x=0, y=150)
ta = Spinbox(raiz, from_=0.0, to=14.9, width=5, increment=0.1)
ta.place(x=150, y=150)
cta = ttk.Combobox(raiz, values=lista, width=5)
cta.place(x=225, y=150)
cta.current(5)

ttk.Button(raiz, text='Cacular', command=analizar).place(x=275, y=200)
raiz.mainloop()
```

3. Aqui procedemos a cargar en vectores los datos introducidos en el campo de texto de nuestra interfaz asi tambion como se procede a cargar todos los datos el archivo csv los cuales nos serviran para compararlos y medir la calidad

```
def analizar():
    newWindows = Tk()
    newWindows.title("Tabla de calidad de vinos")
    df = pd.read_csv(r"winequality-red.csv",sep=';')
    lista = [list(row) for row in df.values]
    similares = {}

    cn = [float(tfa.get()), float(tva.get()), float(tca.get()), float(trs.get()), float(tc.get()), float(tfsd.get()),
          float(ttsd.get()), float(td.get()), float(tph.get()), float(ts.get()), float(ta.get())]
    mini = [4.6, 0.12, 0, 0.9, 0.012, 1, 6, 0.99, 2.74, 0.33, 8.4]
    maxi = [15.9, 1.58, 1.0, 13.9, 0.611, 72.0, 289.0, 1.0, 4.01, 2.0, 14.9]
    weight = [float(ctfa.get()), float(ctva.get()), float(ctca.get()), float(ctrs.get()), float(ctc.get()),
              float(ctfsd.get()),
              float(cttsd.get()), float(ctd.get()), float(ctph.get()), float(cts.get()), float(cta.get())]
```

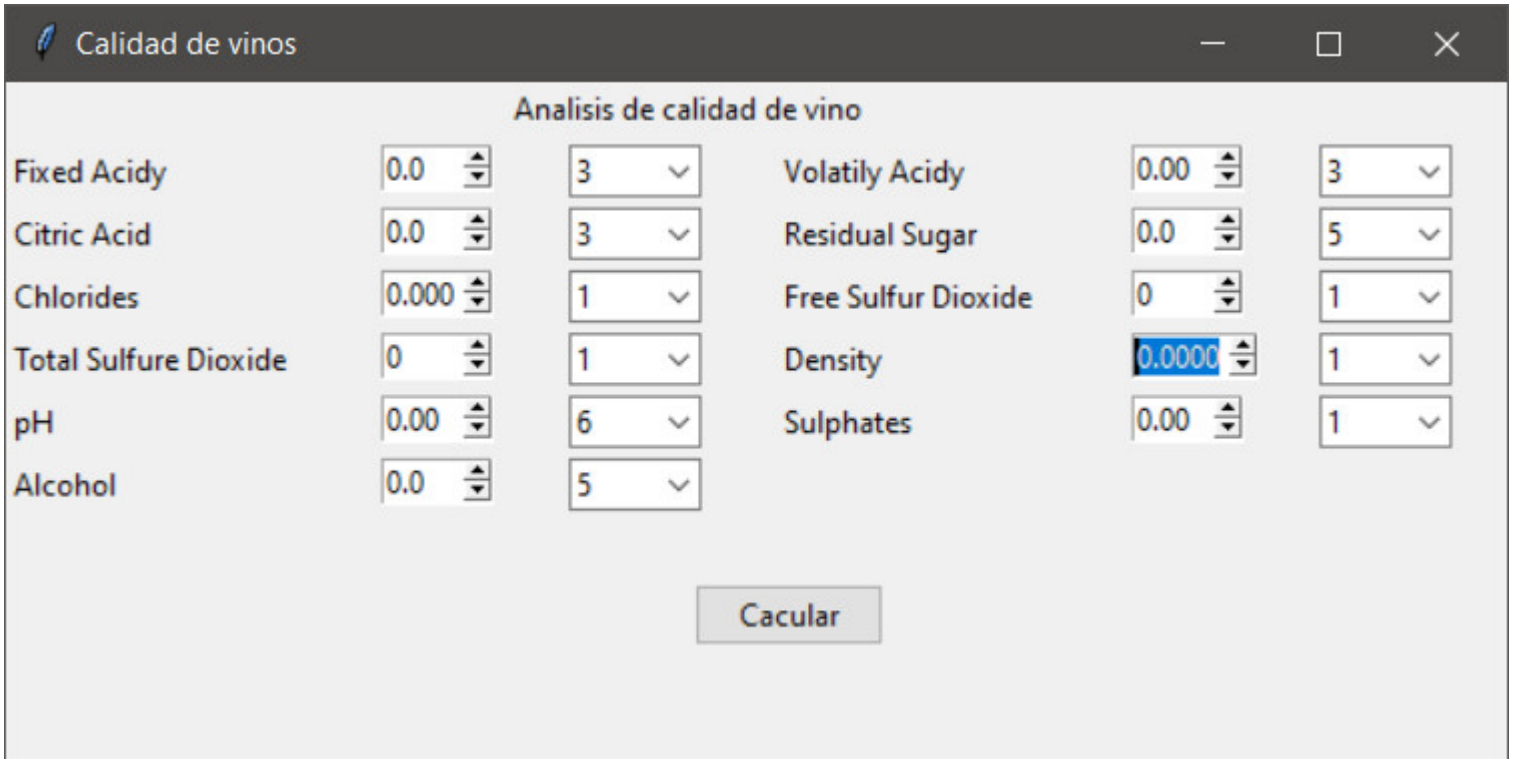
4. En la siguiente lineas de codigos se definira una funcion para nosotros poder realizar el calculo mediante la formula de wguillen y la comparacion con nuestros datos antes alamacenados en una lista que los extraimos de nuestro archivo csv.

```
def similarity(ce):
    valor = 0
    for i in range(len(mini)):
        valor += weight[i] * (1 - ((abs(ce[i] - cn[i])) / (maxi[i] - mini[i])))
    return valor / sum(weight)

for i in range(len(lista)):
    fila = []
    fila = lista[i]
    x = similarity(fila)
    similares.update({str(i): round(x, 3)})

ordenados = dict(sorted(similares.items(), key=operator.itemgetter(1)))
cols = (
"#Wine", "Fixed Acidity", "Volatile Acidity", "Citric Acid", "Residual Sugar", "Chlorides", "Free Sulfure Dioxide"
"Total Sulfure Dioxide", "Density", "pH", "Sulphates", "Alcohol", "Quality", "Similarity")
tree = ttk.Treeview(newWindows, columns=cols, show='headings')
vsb = ttk.Scrollbar(newWindows, orient="vertical", command=tree.yview)
vsb.pack(side=RIGHT, fill=BOTH)

tree.configure(yscrollcommand=vsb.set)
for i in range(len(cols)):
    tree.heading(cols[i], text=cols[i])
    tree.column(cols[i], minwidth=0, width=50)
tree.pack(expand=YES, fill=BOTH)
tam = len(ordenados)
for i in range(tam):
    pos = int(list(ordenados.items())[i][0])
    c1 = lista[int(pos)][0]
    c2 = lista[int(pos)][1]
    c3 = lista[int(pos)][2]
    c4 = lista[int(pos)][3]
    c5 = lista[int(pos)][4]
    c6 = lista[int(pos)][5]
    c7 = lista[int(pos)][6]
    c8 = lista[int(pos)][7]
    c9 = lista[int(pos)][8]
    c10 = lista[int(pos)][9]
    c11 = lista[int(pos)][10]
    c12 = lista[int(pos)][11]
    sim = str(list(ordenados.items())[i][1])
    tree.insert("", 0, i, values=(str(pos), c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11, c12, sim))

# print("Item Mas Similar")
fpos = list(ordenados.items())[tam - 1][0]
```

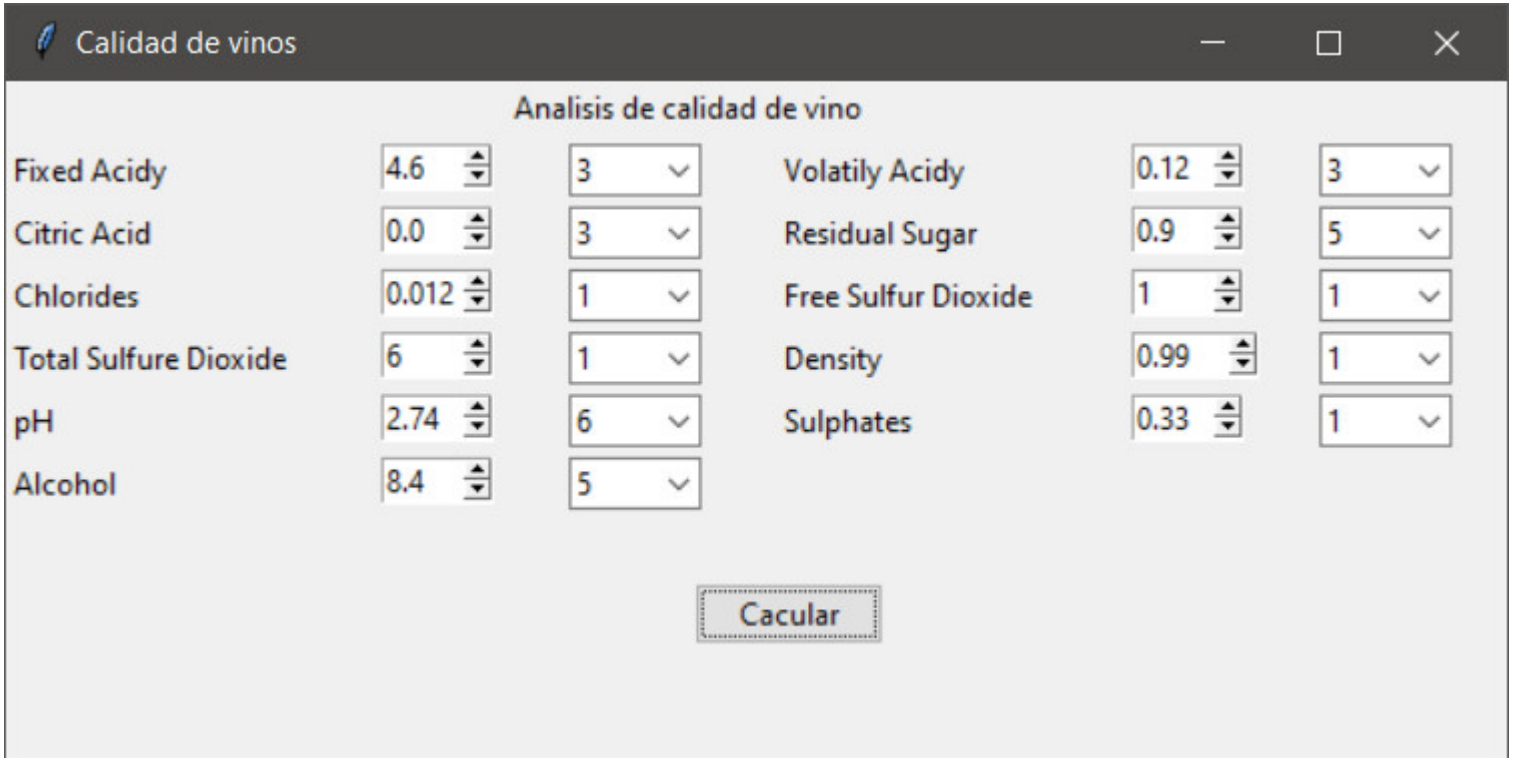
```
fval = list(ordenados.items())[tam - 1][1]
res = lista[int(fpos)][11]
messagebox.showinfo(message="Calidad= " + str(res))
```

Resultados Obtenidos

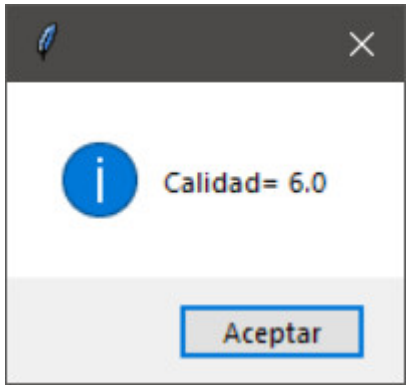
1. Interfaz inicial de nuestra aplicacion.



2. Pruebas meiante los datos dados en el ejemplo de wguillen.



3. Resultado obtenido para la calidad del vino mediante los datos ingresados.



4. Tabla de similiduta en orden de mayor a menor.

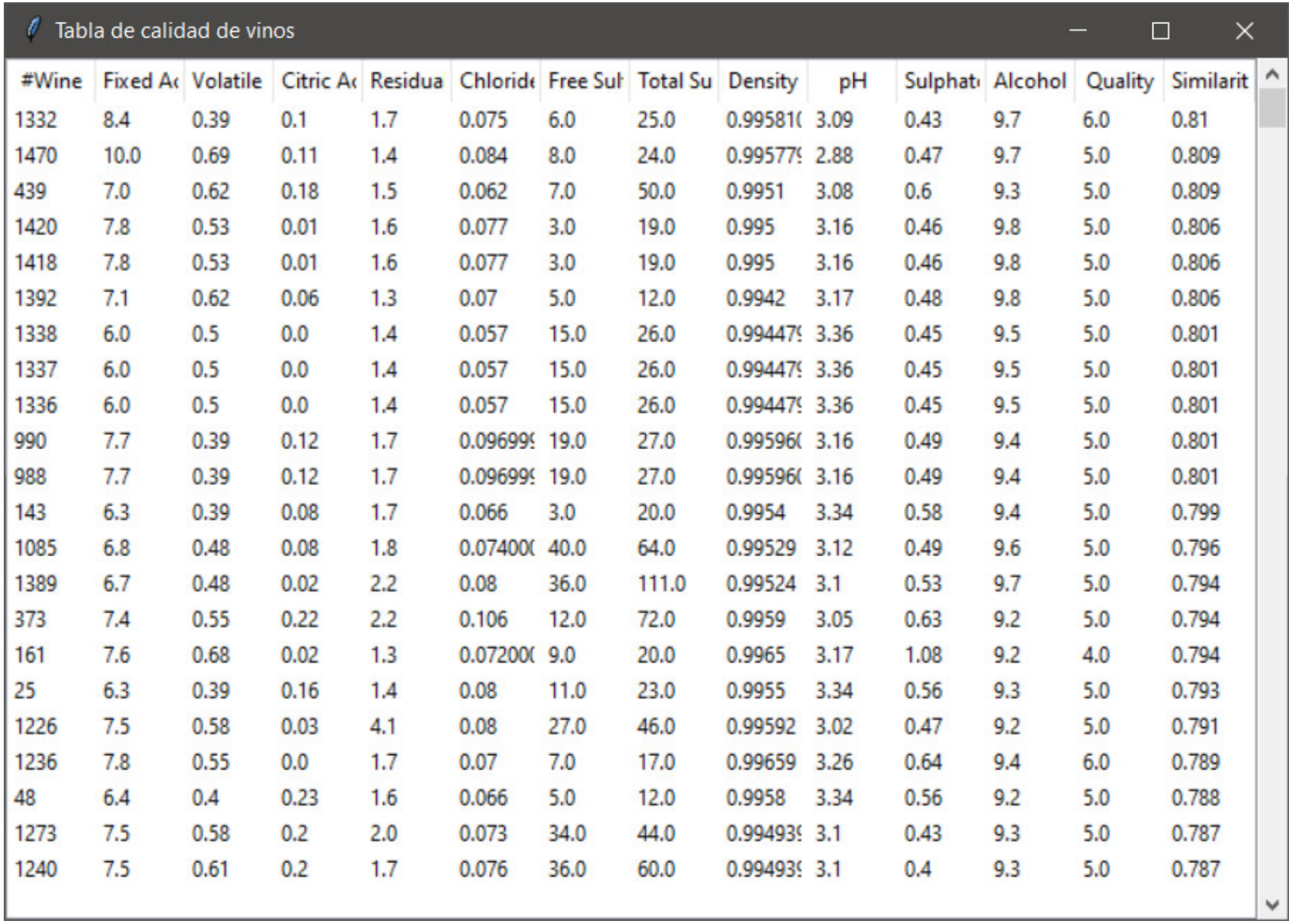


Tabla de calidad de vinos

#Wine	Fixed Ac	Volatile	Citric Ac	Residua	Chloride	Free Sul	Total Su	Density	pH	Sulphat	Alcohol	Quality	Similarit
1332	8.4	0.39	0.1	1.7	0.075	6.0	25.0	0.99581(	3.09	0.43	9.7	6.0	0.81
1470	10.0	0.69	0.11	1.4	0.084	8.0	24.0	0.99577	2.88	0.47	9.7	5.0	0.809
439	7.0	0.62	0.18	1.5	0.062	7.0	50.0	0.9951	3.08	0.6	9.3	5.0	0.809
1420	7.8	0.53	0.01	1.6	0.077	3.0	19.0	0.995	3.16	0.46	9.8	5.0	0.806
1418	7.8	0.53	0.01	1.6	0.077	3.0	19.0	0.995	3.16	0.46	9.8	5.0	0.806
1392	7.1	0.62	0.06	1.3	0.07	5.0	12.0	0.9942	3.17	0.48	9.8	5.0	0.806
1338	6.0	0.5	0.0	1.4	0.057	15.0	26.0	0.99447	3.36	0.45	9.5	5.0	0.801
1337	6.0	0.5	0.0	1.4	0.057	15.0	26.0	0.99447	3.36	0.45	9.5	5.0	0.801
1336	6.0	0.5	0.0	1.4	0.057	15.0	26.0	0.99447	3.36	0.45	9.5	5.0	0.801
990	7.7	0.39	0.12	1.7	0.09699	19.0	27.0	0.99596(	3.16	0.49	9.4	5.0	0.801
988	7.7	0.39	0.12	1.7	0.09699	19.0	27.0	0.99596(	3.16	0.49	9.4	5.0	0.801
143	6.3	0.39	0.08	1.7	0.066	3.0	20.0	0.9954	3.34	0.58	9.4	5.0	0.799
1085	6.8	0.48	0.08	1.8	0.07400(	40.0	64.0	0.99529	3.12	0.49	9.6	5.0	0.796
1389	6.7	0.48	0.02	2.2	0.08	36.0	111.0	0.99524	3.1	0.53	9.7	5.0	0.794
373	7.4	0.55	0.22	2.2	0.106	12.0	72.0	0.9959	3.05	0.63	9.2	5.0	0.794
161	7.6	0.68	0.02	1.3	0.07200(	9.0	20.0	0.9965	3.17	1.08	9.2	4.0	0.794
25	6.3	0.39	0.16	1.4	0.08	11.0	23.0	0.9955	3.34	0.56	9.3	5.0	0.793
1226	7.5	0.58	0.03	4.1	0.08	27.0	46.0	0.99592	3.02	0.47	9.2	5.0	0.791
1236	7.8	0.55	0.0	1.7	0.07	7.0	17.0	0.99659	3.26	0.64	9.4	6.0	0.789
48	6.4	0.4	0.23	1.6	0.066	5.0	12.0	0.9958	3.34	0.56	9.2	5.0	0.788
1273	7.5	0.58	0.2	2.0	0.073	34.0	44.0	0.99493	3.1	0.43	9.3	5.0	0.787
1240	7.5	0.61	0.2	1.7	0.076	36.0	60.0	0.99493	3.1	0.4	9.3	5.0	0.787

Conclusiones

► Mediante estos tipos de algoritmos se puede llegar a crear sistemas expertos muy útiles ya que mediante la comparación y una base de datos bien definida puede ayudar a comparar diferentes productos o consultas dadas por la similitud de las mimas.

► Se determina que los sistemas expertos basados en casos son muy útiles y tienen un gran rendimiento ya que con ayuda de casos previamente registrados podemos solucionar un problema, en este caso determinar la calidad de un vino.

► Se recomienda que las bases de datos de comparación se puedan seguir actualizando ya que con eso se podrá tener más exactitud al realizar los cálculos y obtener comparaciones más precisas

Conclusiones

► https://github.com/wguilen/red-wine-quality-cbr/tree/master/presentation

► https://archive.ics.uci.edu/ml/datasets/Wine+Quality

In [ ]:
```
1
```