Facultad de Ciencias de la Alimentación
Universidad Nacional del Entre Ríos

Inteligencia Artificial

Trabajo practico

Aprendizaje

Angel Gabriel Cenciarini

Yasimel Joaquin Cabello

1. Regresión Lineal simple.

Crear un modelo de regresión lineal en Python, que permita estimar la resistencia delhormigón en función a los diferentes factores existentes dentro de ConcreteStrengthData.csv.

La descripción de los campos contenidos dentro del archivo es la siguiente:

- Componente de cemento: cantidad de cemento que se mezcla.
- BlastFurnaceSlag: - Se mezcla la cantidad de escoria de alto horno
- FlyAshComponent: - Se mezcla la cantidad de FlyAsh
- Componente de agua: - Cantidad de agua que se mezcla
- Componente superplastificante: - Se mezcla la cantidad de superplastificante
- CoarseAggregateComponent: - Se mezcla la cantidad de agregado grueso
- FineAggregateComponent: - Se mezcla la cantidad de agregado grueso
- AgeInDays:- Cuántos días se dejó seco
- Resistencia:- ¿Cuál fue la resistencia final del hormigón? (Objetivo)

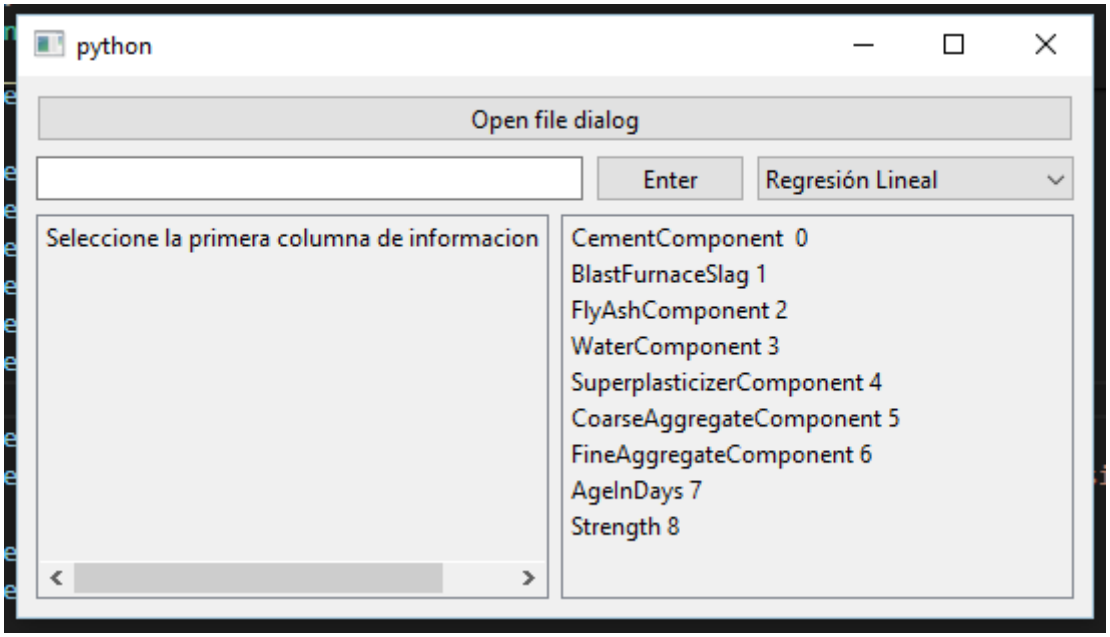
a) Mostrar la información contenida en el dataset. Seleccionar las columnas dedatos necesarias para implementar la regresión lineal.
b) Utilizar los datos de las columnas seleccionadas para crear el modelo deregresión lineal.
c) Separar el dataset original en dos porciones correspondiente a entrenamientoy testeo. Utilizar el 20% de los datos para testeo.
d) Graficar el modelo obtenido, contrastándo las predicciones con las muestrasde testeo.

e) ¿Es adecuado utilizar regresión lineal para este conjunto de datos? ¿Por qué?.

Respuestas:

a) Mostrar la información contenida en el dataset. Seleccionar las columnas de datos necesarias para implementar la regresión lineal.

En este caso, se considera que el usuario ya conoce el tipo de información contenida en el dataset o que puede inferir este conocimiento a partir del nombre de las columnas de este. Por esto mismo se muestran solamente los nombres de todas las columnas. Esto se muestra de la siguiente manera:



Este es el UI del programa, en la columna derecha se pueden observar el nombre de todas las columnas contenidas en el data set seleccionado.

b) Utilizar los datos de las columnas seleccionadas para crear el modelo de regresión lineal.

El programa cuenta con dicho control, al tratarse de una regresión lineal simple solo se admiten dos columnas como entrada. El análisis se realiza con la información de solamente

esas columnas.

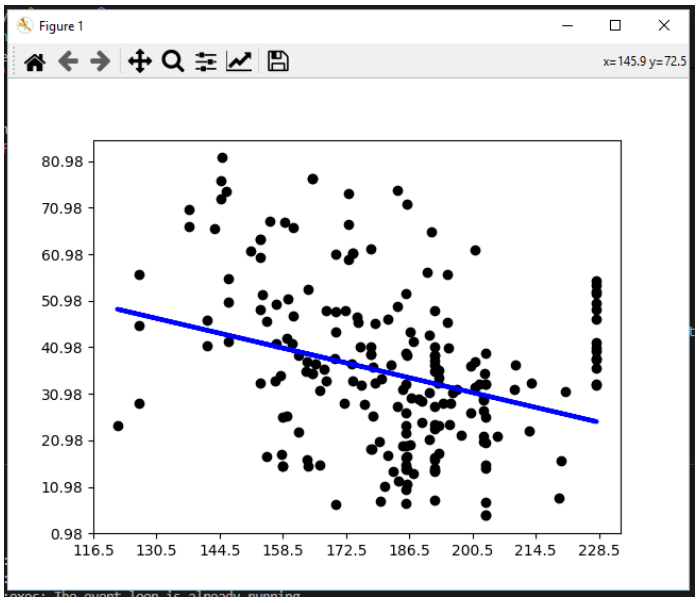
c) Separar el dataset original en dos porciones correspondiente a entrenamiento y testeo. Utilizar el 20% de los datos para testeo.

El programa hace uso de la función *train_test_split()* para elaborar los conjuntos de entrenamiento y prueba a partir de las columnas del data set seleccionadas. Esta función pertenece a la librería *sklearn* que utilizamos para realizar la regresión lineal.
En este caso, se le suministran los siguientes parámetros para que funcione:

- *Columna X:* Data Frame del documento .csv que contiene todos los valores de la columna seleccionada como tal.
- *Columna Y:* Data Frame del documento .csv que contiene todos los valores de la columna que deseamos predecir.
- *Test size:* Porcentaje de los datos que deseamos utilizar para testear el entrenamiento de nuestro sistema. Por defecto, la cantidad sobrante formara parte del conjunto de datos de entrenamiento.

d) Graficar el modelo obtenido, contrastando las predicciones con las muestras de testeo.

Para graficar los resultados del algoritmo se hace uso de la librería *matplotlib.pyplot.* A continuación, se adjunta uno de los posibles resultados para la regresión lineal simple. Como se consideran solo dos columnas, si se quieren analizar todas las columnas se deben realizar múltiples análisis e ir contrastando todas ellas.



En esta imagen podemos observar el resultado de predecir la resistencia del cemente en función de la cantidad de agua que se le puso a este.

e) ¿Es adecuado utilizar regresión lineal para este conjunto de datos? ¿Por qué?

No considero que este algoritmo sea adecuado para intentar predecir la dureza que va a tener el concreto fabricado, debido a que ninguno de los análisis arroja un resultado que nos haga pensar que existe una relación entre alguno de los ítems contenidos y la resistencia del concreto, aun cuando nosotros sabemos que esa relación existe.
En estos casos, donde parece que la información analizada es insuficiente para determinar una respuesta correcta, se hace uso de otras herramientas de predicción como puede ser la *regresión lineal múltiple* que veremos a continuación.

2. Regresión Lineal Múltiple.

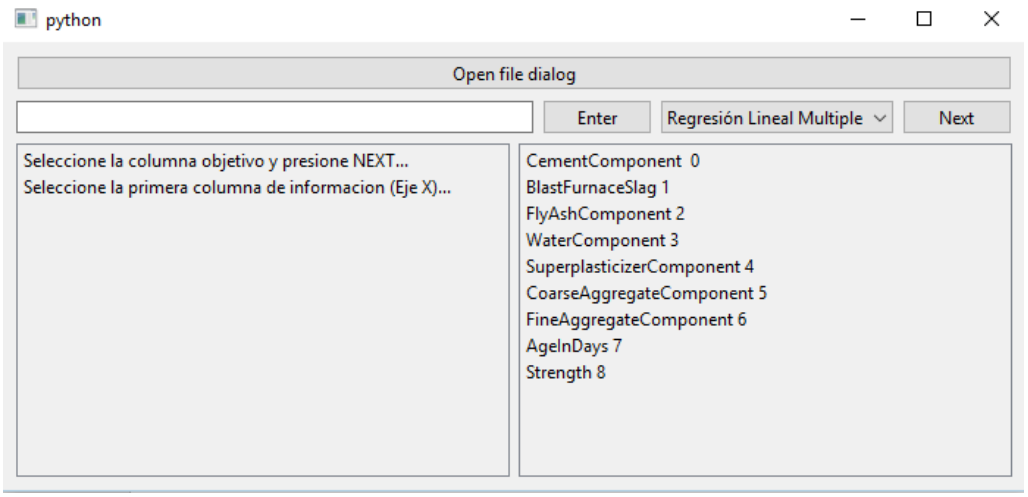
Crear un modelo de regresión lineal múltiple en Python, que permita estimar la resistencia del hormigón en función a los diferentes factores existentes dentro de ConcreteStrengthData.csv.

a) Seleccionar las columnas de datos necesarias para implementar la regresión lineal múltiple. Tener en cuenta que para la selección se debe de tener en cuenta los valores que arroje la matriz de correlación.
b) Utilizar los datos de las columnas seleccionadas para crear el modelo de regresión lineal múltiple.
c) Separar el dataset original en dos porciones correspondiente a entrenamientoy testeo. Utilizar el 20% de los datos para testeo.
d) Graficar el modelo obtenido, contrastándo las predicciones con las muestrasde testeo.
e) ¿Es adecuado utilizar regresión lineal múltiple para el conjunto de datos seleccionados? ¿Por qué?

Respuestas:

a) Seleccionar las columnas de datos necesarias para implementar la regresión lineal múltiple. Tener en cuenta que para la selección se debe de tener en cuenta los valores que arroje la matriz de correlación.

Al igual que en el punto anterior, la UI del programa nos señala todas las columnas que existen en la base de datos. La única diferencia es que en este caso tenemos un botón extra *Next*, este botón es necesario ya que para la regresión lineal múltiple se deben seleccionar todas las columnas que conformaran nuestra *Columna X*.



b) Utilizar los datos de las columnas seleccionadas para crear el modelo de regresión lineal múltiple.

En este caso, al tratarse de un modelo de regresión múltiple se debe tener en cuenta que debemos utilizar todas las columnas seleccionadas para el análisis y realizar nuestro código en consecuencia. El botón Next sirve para continuar seleccionando columnas hasta que tengamos todas las que creemos necesarias, luego de lo cual presionamos Enter para empezar con el análisis.

c) Separar el dataset original en dos porciones correspondiente a entrenamiento y testeo. Utilizar el 20% de los datos para testeo.

El programa hace uso de la función *train_test_split()* para elaborar los conjuntos de entrenamiento y prueba a partir de las columnas del data set seleccionadas. Esta función pertenece a la librería *sklearn* que utilizamos para realizar la regresión lineal.
En este caso, se le suministran los siguientes parámetros para que funcione:

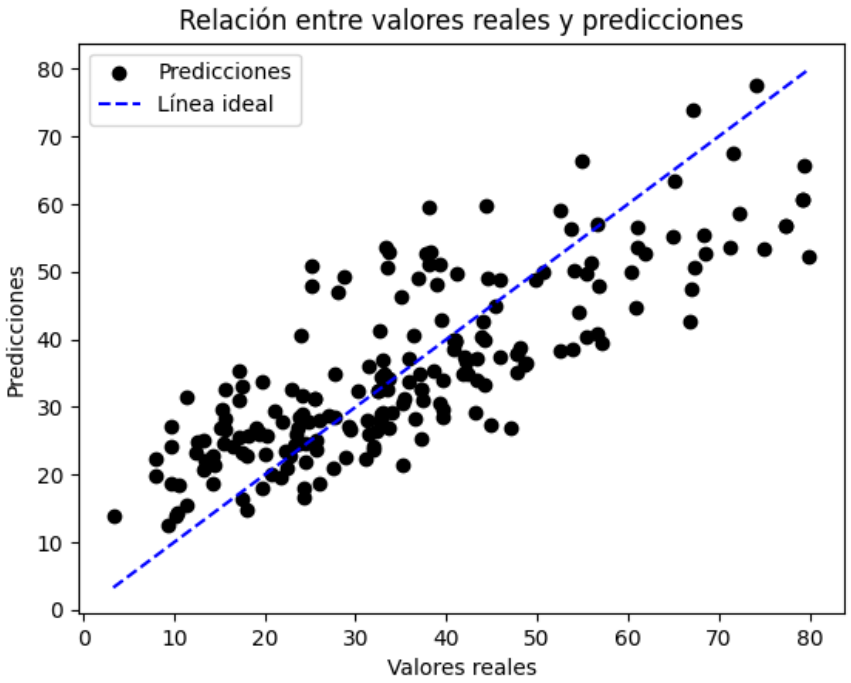
- *Columna X:* Data Frame del documento .csv que contiene todos los valores de las múltiples columnas seleccionadas.
- *Columna Y:* Data Frame del documento .csv que contiene todos los valores de la

columna que deseamos predecir.

- *Test size:* Porcentaje de los datos que deseamos utilizar para testear el entrenamiento de nuestro sistema. Por defecto, la cantidad sobrante formara parte del conjunto de datos de entrenamiento.

d) Graficar el modelo obtenido, contrastando las predicciones con las muestras de testeo.

Para graficar los resultados del algoritmo se hace uso de la librería *matplotlib.pyplot*. En este caso, en la siguiente imagen se muestra la predicción de la resistencia del concreto debido a la influencia de todas las variables registradas en las demás columnas de la base de datos.



e) ¿Es adecuado utilizar regresión lineal múltiple para el conjunto de datos seleccionados? ¿Por qué?

Considero que en para este caso es mejor utilizar regresión lineal múltiple en lugar de

su contraparte simple, ya que a partir de los resultados entregados por el algoritmo se confirma nuestra teoría de que existe relación entre los componentes del concreto y la resistencia de este. Al tener la posibilidad de elegir más variables, es más sencillo encontrar relaciones entre estas y nuestra característica a predecir.

3. Regresión Logística.

Crear un modelo de Regresión Logística utilizando el dataset ai4i2020.csv que cuenta con datos de mantenimiento predictivo encontrados en la industria. El conjunto de datos consta de 10 000 puntos de datos almacenados como filas con 14 característicasen columnas. Dentro de estas características se encuentran datos de Temperatura del proceso, par, velocidad de rotación etc, y datos de fallas clasificados con 0 si no existefalla de ese tipo y con 1 si falla.

a) Mostrar la información contenida en el dataset. Seleccionar las columnas de datos necesarias para implementar la regresión Logística.
b) Teniendo cuenta las características que deben cumplir los datos para poder aplicar regresión logística, hacer uso de la siguiente línea de código para reconocer las relaciones entre variables (Columnas de datos):

```
data_frames[['Nombre_columna1','Nombre_columna2"]].plot.scatter(x='Nombr
e_columna1',y='Nombre_columna2')
```

c) Implemente un modelo de regresión logística para cada uno de los tipos de fallas. Seleccione los predictores y de ser necesario escale los mismos. Recuerde que se puede escalar las variables cuando son completamente distintas en magnitudes, unidades y rango.
d) Graficar y comparar los datos que predice el modelo con los datos de testeo.Para poder graficar a continuación se deja un fragmento de código.

```
# Se crea una ventana grafica para comparar los datos de testeo
#con los de la predicción
import tkinter as tk
ventana = tk.Tk()
ventana.geometry("600x300")

frame = tk.Frame(ventana)
frame.pack(padx=10, pady=10)

lista_datos1 = tk.Listbox(frame)
lista_datos1.pack(side=tk.LEFT, fill=tk.BOTH)

lista_datos2 = tk.Listbox(frame)
lista_datos2.pack(side=tk.LEFT, fill=tk.BOTH)

datos1 = y_test
datos2 = y_pred

for i, dato in enumerate(datos1, start=1):
```

```
    color = "red" if dato == 0 else "green"
    lista_datos1.insert(tk.END, f"{dato} ({i})")
    lista_datos1.itemconfig(tk.END, bg=color)

for i, dato in enumerate(datos2, start=1):
    color = "red" if dato == 0 else "green"
    lista_datos2.insert(tk.END, f"{dato} ({i})")
    lista_datos2.itemconfig(tk.END, bg=color)

ventana.mainloop()
```

e) Para cada uno de los tipos de falla, utilizando los métodos disponible ensklearn.metrics:
   a. Armar la matriz de confusión.
   b. Evaluar la precisión del modelo.
   c. Evaluar la exactitud del modelo con accurancy.

f) En base al punto anterior, analice y conteste:

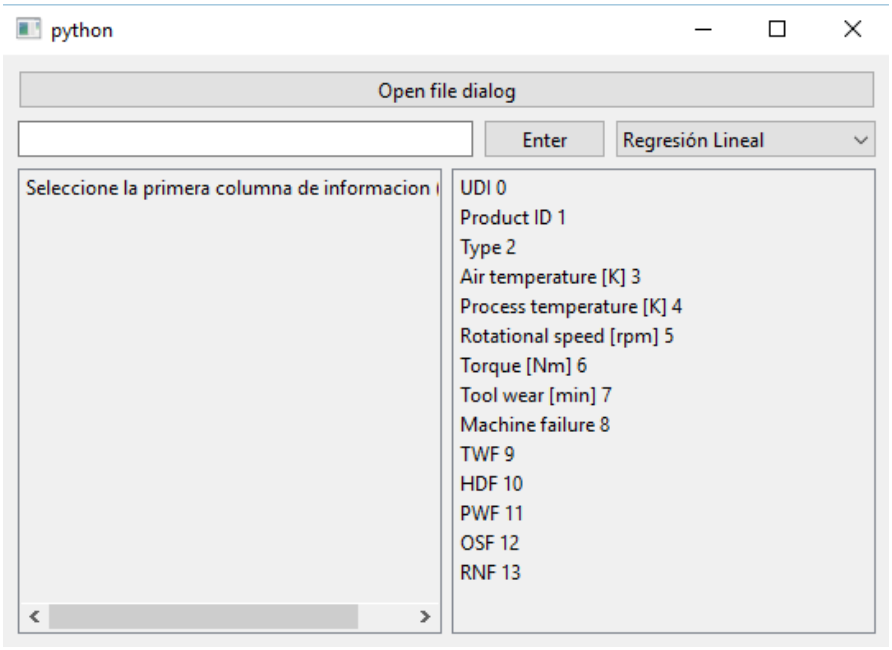
i. ¿Cómo se comporta el modelo si consideramos todos los predictores?

ii. ¿Qué sucede cuando solo consideramos como predictores algunos de ellos?

Respuestas:

a) Mostrar la información contenida en el dataset. Seleccionar las columnas de datos necesarias para implementar la regresión Logística.

Al igual que en el punto anterior, la UI del programa nos señala todas las columnas que existen en la base de datos. La única diferencia es que en este caso tenemos un botón extra Next, este botón nos permite realizar la predicción sobre más de una columna en un mismo análisis. De esta forma, si tenemos distintos tipos de fallas como ocurre en el data set seleccionado, vamos a poder predecir todas ellas y no será necesario realizar los análisis uno por uno.



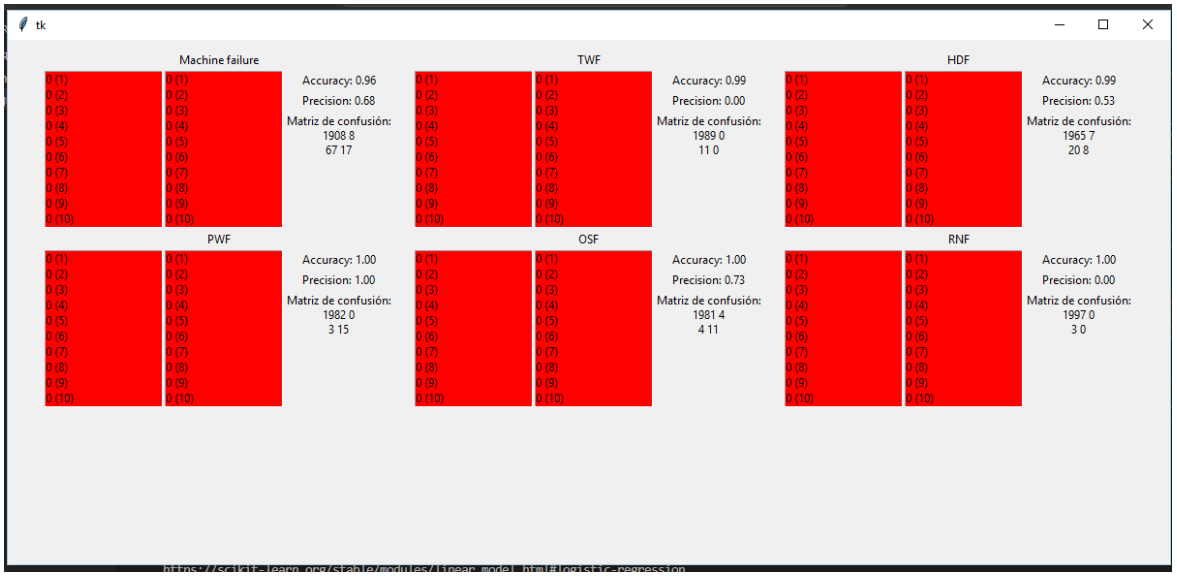
c) Implemente un modelo de regresión logística para cada uno de los tipos de fallas. Seleccione los predictores y de ser necesario escale los mismos. Recuerde que se puede escalar las variables cuando son completamente distintas en magnitudes, unidades y rango.

El programa hace uso de la función *train_test_split()* para elaborar los conjuntos de entrenamiento y prueba a partir de las columnas del data set seleccionadas. Esta función pertenece a la librería *sklearn* que utilizamos para realizar la regresión lineal. En este caso, se le suministran los siguientes parámetros para que funcione:

- *Columna X:* Data Frame del documento .csv que contiene todos los valores de las múltiples columnas seleccionadas.
- *Columna Y:* Data Frame del documento .csv que contiene todos los valores de la columna que deseamos predecir.
- *Test size:* Porcentaje de los datos que deseamos utilizar para testear el entrenamiento de nuestro sistema. Por defecto, la cantidad sobrante formara parte del conjunto de datos de entrenamiento.

d) Graficar y comparar los datos que predice el modelo con los datos de testeo.

Para graficar los resultados obtenidos con el algoritmo se hace uso de la librería *tkimter* que nos permite poner listas en una ventana, de esta forma podemos ver todos los predichos comparados con los datos reales, observando de mejor manera si nuestro algoritmo puede predecir de forma correcta.
En este caso, se agregó un segmento de código que nos permite visualizar de forma matricial todos los datos analizados, de esta forma podemos observar todos los análisis realizados independientemente de su número.



e) Para cada uno de los tipos de falla, utilizando los métodos disponible ensklearn.metrics:
   a. Armar la matriz de confusión.
   b. Evaluar la precisión del modelo.
   c. Evaluar la exactitud del modelo con accurancy.

Para mayor comodidad, y como se ve en la imagen del inciso anterior, estos valores estaedisticos que dan idea de la precisión del algoritmo se muestran en pantalla junto con los resultados. Esto también es modular y va enlazado a cada análisis hecho, en este caso, se realiza de la siguiente manera:

```
accuracy = accuracy_score(Ytest, y_pred)
accuracy_label = tk.Label(subframe, text=f"Accuracy: {accuracy:.2f}")
accuracy_label.pack()

precision = precision_score(Ytest, y_pred)
precision_label = tk.Label(subframe, text=f"Precision: {precision:.2f}")
precision_label.pack()

conf_matrix = confusion_matrix(Ytest, y_pred)
conf_matrix_str = "\n".join(" ".join(map(str, row)) for row in conf_matrix)
conf_matrix_label = tk.Label(subframe, text=f"Matriz de confusión:\n{conf_matrix_str}")
conf_matrix_label.pack()
```

4. K vecinos más cercanos.

Contamos con el dataset Occupancy_Estimation.csv que contiene los datos para estimar el número exacto de ocupantes en una habitación utilizando múltiples sensores ambientales no intrusivos como temperatura, luz, sonido, CO2 y PIR. Los datos fueron recolectados por un período de 4 días de manera controlada con la ocupación en la habitación variando entre 0 y 3 personas.

Utilizar un clasificador K vecinos más cercanos para predecir el grupo de los casos desconocidos o de testeo.

a) Construir un programa Python para resolver el ejercicio. De ser necesario utilizar sklearn.preprocessing. para preparar los datos.
b) Separar el dataset original en dos porciones correspondiente a entrenamiento y testeo. Tener en cuenta que debo usar los mismos datos para todos los clasificadores.
c) Construir el clasificador KNN con 7 vecinos más cercanos.
   Utilizar: from sklearn.neighbors import KNeighborsClassifier
d) Obtener una versión alternativa del clasificador que difieran en la forma de laimportancia relativa de los vecinos conforme a la distancia.
e) Utilizar acurracy como métrica de comparación entre ambos clasificadores. ¿cuál es más preciso?

Respuestas:

a) Construir un programa Python para resolver el ejercicio. De ser necesario utilizar sklearn.preprocessing. para preparar los datos.

El programa hace uso de la función *StandarScaler()* de la librería *Sklearn.preprocessing* para escalar los datos, preparando estos para su uso en el algoritmo de inteligencia artificial.

```
escalar = StandardScaler()
Xtrain = escalar.fit_transform(Xtrain)
Xtest = escalar.transform(Xtest)
```

b) Separar el dataset original en dos porciones correspondiente a entrenamiento y testeo. Tener en cuenta que debo usar los mismos datos para todos los clasificadores.

El programa hace uso de la función *train_test_split()* para elaborar los conjuntos de entrenamiento y prueba a partir de las columnas del data set seleccionadas. Esta función pertenece a la librería *sklearn* que utilizamos para realizar la regresión lineal.
En este caso, se le suministran los siguientes parámetros para que funcione:

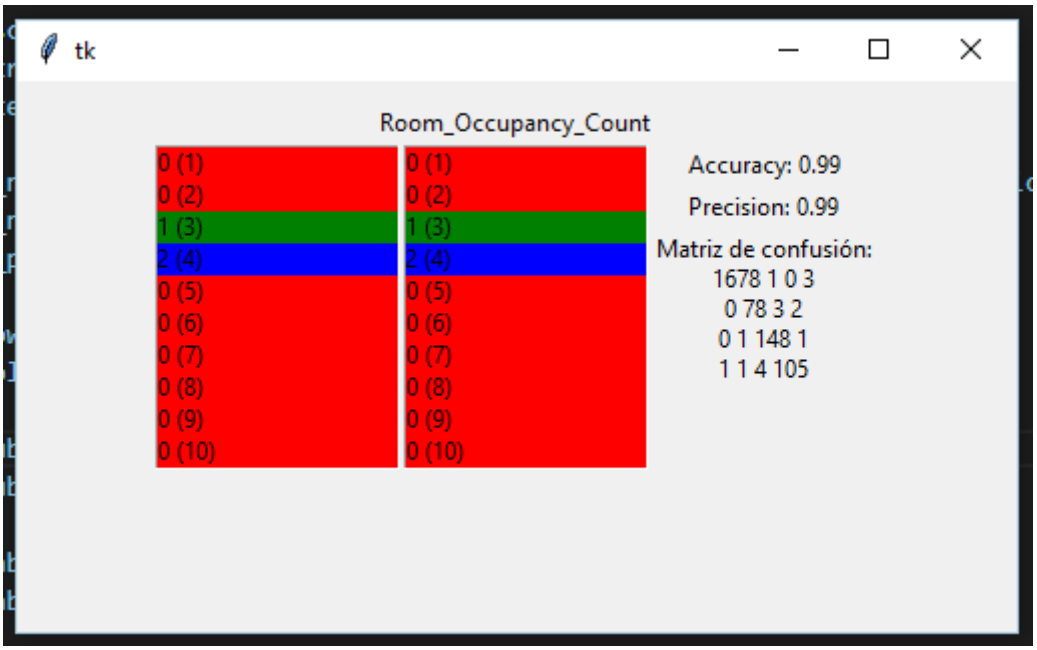
- *Columna X:* Data Frame del documento .csv que contiene todos los valores de las múltiples columnas seleccionadas.
- *Columna Y:* Data Frame del documento .csv que contiene todos los valores de la columna que deseamos predecir.
- *Test size:* Porcentaje de los datos que deseamos utilizar para testear el entrenamiento de nuestro sistema. Por defecto, la cantidad sobrante formara parte del conjunto de datos de entrenamiento.

c) Construir el clasificador KNN con 7 vecinos más cercanos. Utilizar: from sklearn.neighbors import KNeighborsClassifier

```
K_nn = KNeighborsClassifier(n_neighbors=7, weights='distance', metric='minkowski', p=2)
K_nn.fit(Xtrain,Ytrain)
y_pred = K_nn.predict(Xtest)
```

e) Utilizar acurracy como métrica de comparación entre ambos clasificadores.

En este caso, no se implementaron variaciones entre los clasificadores, pero al igual que en el punto anterior se creó un sistema que distribuye de forma matricial todos los resultados para que podamos analizar todos los datos que deseemos y que estos sigan siendo representados de forma correcta y cómoda.



Además de la representación grafica se agregó una matriz de confunsión y distintos parámetros de precisión y exactitud que dan una idea de la calidad del entrenamiento que posee nuestro algoritmo.