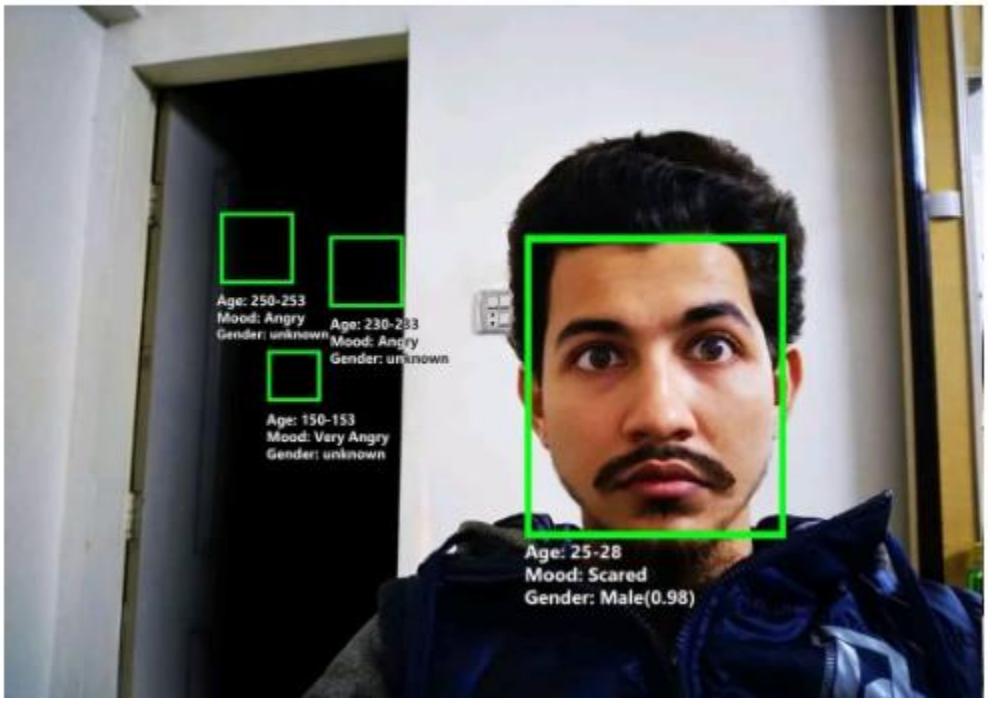
Supervised Learning

by Ivan Alduoln



In [134]:
```
# Archivo Heart Attack.csv - ¿Cuales son los factores que pueden incrementar o disminuir la probabilidad de un ataque al corazón

import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M25 DS/Avance de Proyecto\PARTE 2/Heart Attack.csv')
df
```

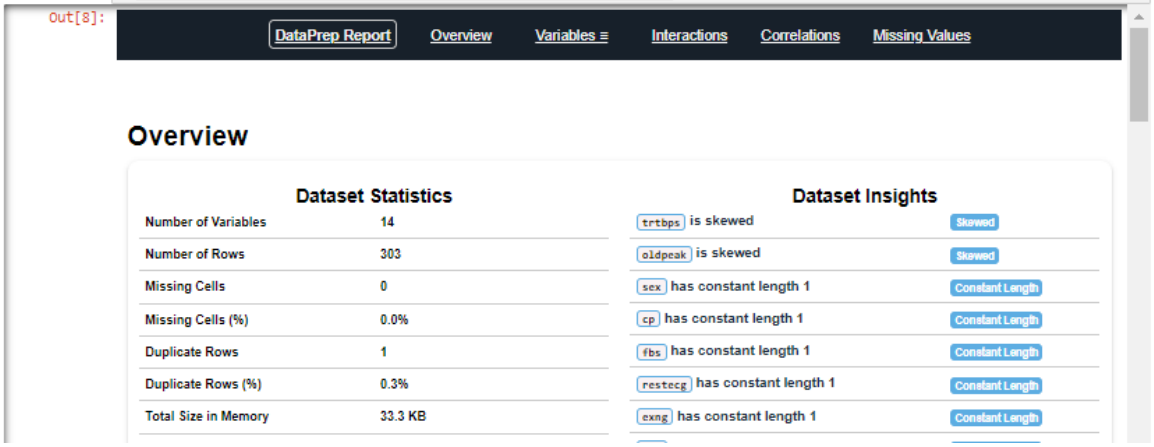
Out[134]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

In [8]:
```
# Hacer EDA (Exploratory Data Analysis) suele ser un tanto laborioso dependiendo del detalle al que se quiera llevar, pero prueb
from dataprep.eda import create_report

create_report(df)
```

Out[8]:



In [9]:
```
#!pip install dataprep
```

k-Nearest Neighbors

Habiendo hecho un Análisis Exploratorio de los factores que pueden o no tener más posibilidad de un ataque al corazón, es hora de crear tu primer clasificador!!! usando el algoritmo de k-NN.

Nota: es importante garantizar que los datos esten en el formato requerido por la librería de `scikit-learn`. La información debe estar en una matriz en la que cada columna sea una variable y cada fila una observación diferente, en este caso, el registro de análisis clinico por paciente. Y la variable objetivo debe ser una sola columna con el mismo número de observaciones.

In [135]:
```
# Importa la librería para un clasificador k-NN de sklearn
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

# Crea dos arreglos "X", "y" que contengan los valores de las variables independientes y la variable objetivo
cols_name = ['age','sex','cp','trtbps','chol','fbs','restecg','thalachh','exng','oldpeak','slp','caa','thall']
x = df[cols_name].values
y = df.output.values

# Crea un clasificador k-NN con 6 vecinos
knn = KNeighborsClassifier(n_neighbors = 6)

# Ajusta el clasificador a las variables
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 1)

# Entrenamos el modelo
knn.fit(x_train, y_train)

# realizamos predicicones

knn_pred = knn.predict(x_test)
```

Predicción

Una vez que entrenamos al clasificador k-NN, ahora lo podemos usar para predecir un nuevo registro. Para este caso, no hay datos sin clasificar disponibles ya que todos se usaron para entrenar al modelo. Para poder calcular una predicción, vamos a usar el método `.predict()` pero, para esto vamos a simular una observación completamente nueva

In [136]:
```
# Crea un arreglo simulando una observación
X_new = {'age': 38,'sex': 0 ,'cp': 1 ,'trtbps': 122 ,'chol': 200 ,'fbs': 1 ,'restecg': 1 ,'thalachh': 110 ,'exng': 0 ,'oldpeak':
X_new_array = np.array([[X_new['age'],X_new['sex'],X_new['cp'],X_new['trtbps'],X_new['chol'],X_new['fbs'],
                         X_new['restecg'],X_new['thalachh'],X_new['exng'],X_new['oldpeak'],X_new['slp'],X_new['caa'],X_new['thall
# Predice la clasificación para el arreglo que creaste
y_new_pred = knn.predict(X_new_array)
print("Prediction: {0}".format(y_new_pred))
```

```
Prediction: [0]
```

In [137]:
```python
# Importa el archivo de MNIST
digits = pd.read_csv('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M25 DS/Avance de Proyecto/PARTE 2/MNIST.csv')
digits
```

Out[137]:

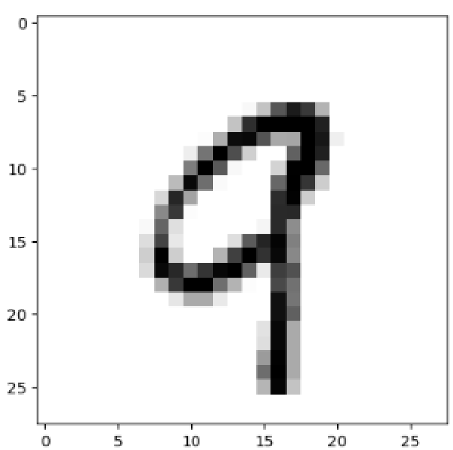
l0	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	...	pixel774	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

columns

In [138]:
```python
# Crea una variable 'cols' para hacer referencia a todas las columnas que contienen la palabra 'pixel'

cols = [col for col in digits.columns if 'pixel' in col]
```

In [139]:
```python
import matplotlib.pyplot as plt
# Vamos a imprimir un digito
i = 41999
print("El número es:", digits.loc[i,'label'])
plt.imshow(digits.loc[i, cols].values.reshape((28,28)).astype(float), cmap=plt.cm.gray_r, interpolation='nearest')
plt.show()
```

El número es: 9



In [140]:
```
# Importa la librería para entrenamiento y prueba de datos y la librería para calcular la precisión
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [141]:
```
# Crea los arreglos para las variables independientes y la variable objetivo
X = digits.drop(['label'], axis = 1)
y = digits.label.values

# Divide los arreglos anteriores en conjuntos de training y test en una proporcion del 70/30
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state= 1)

# Instancia un clasificador k-NN con 14 vecinos
knn = KNeighborsClassifier(n_neighbors = 14)

# Ajusta (Entrenamiento) el clasificador en el conjunto de entrenamiento
knn.fit(X_train, y_train)

# Calcular las predicciones sobre el conjunto de prueba
y_pred = knn.predict(X_test)

# Verificar la precisión del modelo
print(accuracy_score(y_test, y_pred))
```

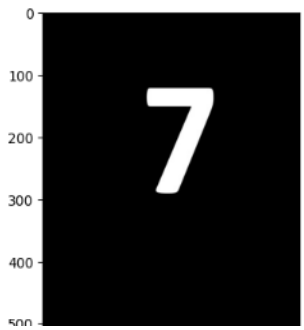
```
0.9607142857142857
```

Reconocimiento de tu imagen

Con todo lo anterior, podemos hacer el reconocimiento de cualquier digito que dibujes, ¿Estás list@?

In [152]:
```
# Vamos a visualizar la imagen de un número que vas a crear en tu computador con la aplicación de paint, ésta imagen debe de ten
image = plt.imread('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M25 DS/Avance de Proyecto/PARTE 2/numero.jpg') # Coloca
plt.imshow(image)
```

Out[152]: `<matplotlib.image.AxesImage at 0x24d32b0de10>`



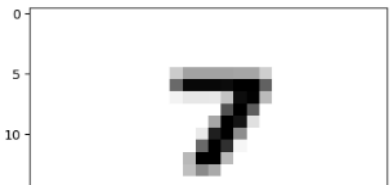
In [153]:
```
# Con esta libreria transformaremos la imagen creada a un formato de 28x28 pixeles
from PIL import Image
pil = Image.open('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M25 DS/Avance de Proyecto/PARTE 2/numero.jpg')
image_resize = pil.resize((28, 28))

# Vamos transformar la nueva imagen en un array donde se almacenara la información de los pixeles

pixels = np.asarray(image_resize)
```

In [154]:
```
# Necesitamos hacer algunas configuraciones a la imagen debido al formato de datos que esta alimentando al modelo y a la configu
arr = pixels.transpose(2, 0, 1).reshape(-1, pixels.shape[1])[0:28]

image_final = arr.ravel().reshape(1, -1)
```

In [155]:
```
# Calcula la predicción del modelo con el número que creaste, ¿Fue la clasificación correcta? :O

knn_pred = knn.predict(image_final)
print("El número es:", knn_pred)
plt.imshow(arr, cmap=plt.cm.gray_r, interpolation='nearest')
```

```
El número es: [1]
```

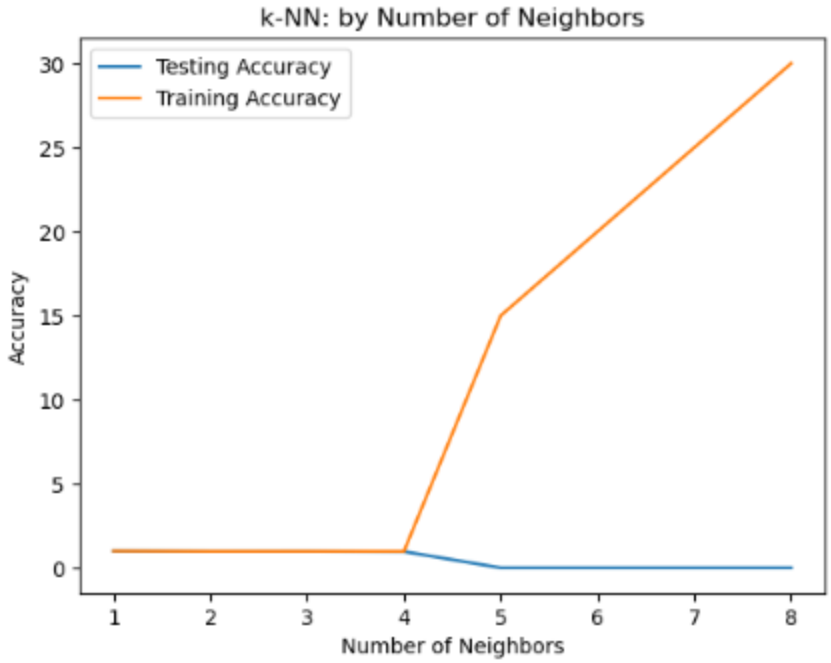
Out[155]: `<matplotlib.image.AxesImage at 0x24d32b97150>`



Overfit and Underfit

¿Cual es mi numero ideal para elegir el parametro *K*? Vamos a calcular los valores de precisión para los conjuntos e entrenamiento y prueba para una rango de valores k. Al observar cómo difieren estos valores podremos observar cual es el mejor parametro sin caer en un *Overfit* o un *Underfit*.

In [123]:
```
# Coniguración de arreglos iniciales
neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop para diferentes valores de k
for i, k in enumerate(neighbors):
    # Clasificador k-NN para el parametro k
    knn = KNeighborsClassifier(n_neighbors=k)

    # Ajuste del clasificador al dataset de entrenamiento
    knn.fit(X_train, y_train)

    # Calculo de precision sobre el dataset de entrenamiento
    train_accuracy[i] = knn.score(X_train, y_train)

    # Calculo de precision sobre el dataset de prueba
    test_accuracy[i] = knn.score(X_test, y_test)

# Grafico para encontrar un valor optimo de k
plt.plot(neighbors, test_accuracy, label = 'Testing Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training Accuracy')
plt.title('k-NN: by Number of Neighbors')
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



Regresión Logística

Haz la predicción de tu imagen, pero esta vez por medio de una Regresión Logística, ¿Cuál de los dos modelos te da mejores resultados?

- El Modelo KNN da un 96% de efectividad
- El modelo LBFGS da un 91.9% de efectividad

In [160]:
```
from sklearn.linear_model import LogisticRegression

# Divide los arreglos anteriores en conjuntos de training y test en una proporcion del 70/30
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state= 1)

# Instancia un clasificador k-NN con 14 vecinos
lb= LogisticRegression(solver = 'lbfgs')

# Ajusta (Entrenamiento) el clasificador en el conjunto de entrenamiento
lb.fit(X_train, y_train)

# Calcular las predicciones sobre el conjunto de prueba
y_pred = lb.predict(X_test)

# Verificar la precisión del modelo
print(accuracy_score(y_test, y_pred))
```

```
0.9192857142857143
```

In [161]:
```
# Vamos a visualizar la imagen de un número que vas a crear en tu computador con la aplicación de paint, ésta imagen debe de ten
image = plt.imread('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M25 DS/Avance de Proyecto/PARTE 2/numero.jpg') # Coloca
plt.imshow(image)

# Con esta libreria transformaremos la imagen creada a un formato de 28x28 pixeles
from PIL import Image
pil = Image.open('C:/Users/Isaac/Desktop/IHD/EBAC DT/CIENCIA DE DATOS/M25 DS/Avance de Proyecto/PARTE 2/numero.jpg')
image_resize = pil.resize((28, 28))

# Vamos transformar la nueva imagen en un array donde se almacenara la información de los pixeles

pixels = np.asarray(image_resize)

# Necesitamos hacer algunas configuraciones a la imagen debido al formato de datos que esta alimentando al modelo y a la configu
arr = pixels.transpose(2, 0, 1).reshape(-1, pixels.shape[1])[0:28]

image_final = arr.ravel().reshape(1, -1)
```

In [162]:
```
# Calcula la predicción del modelo con el número que creaste, ¿Fue la clasificación correcta? :O

y_pred = lb.predict(image_final)
print("El número es:", y_pred)
plt.imshow(arr, cmap=plt.cm.gray_r, interpolation='nearest')
```

```
El número es: [7]
```

Out[162]:
```
<matplotlib.image.AxesImage at 0x24d3385f250>
```