1 Introducción:

En este proyecto se creará un modelo que pueda clasificar y generar dígitos a partir de la base de datos de MNIST. Esta base de datos contiene imágenes de dígitos escritos a mano y sus respectivas etiquetas. Se puede acceder a estas imágenes y etiquetas desde el repositorio de github, en el mismo lugar donde se encuentra este documento.

El modelo se fundamentará en la metodología Naive Bayes, la cual es una forma de clasificar cosas usando la probabilidad. Naive Bayes usa la probabilidad para predecir a qué grupo o categoría pertenece algo, basándose en la información que tiene. Por ejemplo, si queremos clasificar un correo electrónico como spam o no spam, Naive Bayes usa la probabilidad de que el correo electrónico contenga ciertas palabras o características que son típicas del spam o del correo normal.

Naive Bayes funciona usando una fórmula matemática llamada teorema de Bayes. Esta fórmula nos dice cómo actualizar la probabilidad de algo cuando tenemos nueva información. Por ejemplo, si sabemos que el 80% de los correos electrónicos son spam, y que el 90% de los correos electrónicos spam contienen la palabra “oferta”, podemos usar el teorema de Bayes para calcular la probabilidad de que un correo electrónico sea spam si contiene la palabra “oferta”. El resultado sería del 96%.

El teorema de Bayes establece que la probabilidad de que una hipótesis H sea verdadera, dado que se ha observado una evidencia E, es igual a la probabilidad de que la evidencia E se observe, dado que la hipótesis H es verdadera, multiplicada por la probabilidad previa de que la hipótesis H sea verdadera, y dividida por la probabilidad de observar la evidencia E. La fórmula del teorema de Bayes es la siguiente:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

En el contexto de la clasificación, la hipótesis H es la clase a la que pertenece un dato, y la evidencia E son las características o atributos del dato. Por ejemplo, si queremos clasificar una imagen de un dígito escrito a mano, la hipótesis H sería el valor del dígito (0, 1, 2, ..., 9) y la evidencia E serían los píxeles de la imagen. El algoritmo de Naive Bayes calcula la probabilidad de cada clase posible, dado el dato, y elige la clase con mayor probabilidad como la predicción.

El algoritmo de Naive Bayes se llama así porque hace una suposición muy simplista: que las características del dato son independientes entre sí, dada la clase. Es decir, que no hay relación entre los píxeles de una imagen, si sabemos que representa un dígito determinado. Esta suposición es ingenua porque en realidad las características suelen estar correlacionadas entre sí. Por ejemplo, si sabemos que una imagen representa un 8, esperamos que los píxeles del centro estén encendidos y los de las esquinas apagados. Sin embargo, el algoritmo de Naive Bayes ignora estas dependencias y trata cada característica por separado.

A pesar de esta suposición ingenua, el algoritmo de Naive Bayes puede funcionar bien en algunos problemas de clasificación, especialmente cuando hay muchas características y los datos están bien balanceados. El algoritmo de Naive Bayes tiene varias ventajas: es simple, rápido, fácil de implementar y requiere pocos parámetros. También puede manejar problemas de clasificación multiclase sin necesidad de combinar varios modelos binarios.

Sin embargo, el algoritmo de Naive Bayes también tiene algunas desventajas: puede ser muy sensible a las características irrelevantes o redundantes, puede sufrir el problema del cero cuando hay valores desconocidos o infrecuentes en los datos y puede tener un rendimiento inferior a otros métodos más sofisticados cuando las características no son independientes entre sí.

En el caso del reconocimiento de imágenes de dígitos, el algoritmo de Naive Bayes puede predecir el valor del dígito a partir de los píxeles de la imagen usando la siguiente fórmula:

$$P(dígito|píxeles) = \frac{P(píxeles|dígito)P(dígito)}{P(píxeles)}$$

Donde:

- $P(dígito|píxeles)$ es la probabilidad posterior de que el dígito sea el correcto, dado los píxeles de la imagen.
- $P(píxeles|dígito)$ es la probabilidad condicional o verosimilitud de observar los píxeles de la imagen, dado que el dígito es el correcto.
- $P(dígito)$ es la probabilidad previa o a priori de que el dígito sea el correcto.
- $P(píxeles)$ es la probabilidad marginal o evidencia de observar los píxeles de la imagen.

Para calcular estas probabilidades se necesita un conjunto de datos etiquetados con imágenes y sus respectivos dígitos. La base de datos contiene 70.000 imágenes en escala de grises de 28x28 píxeles cada una, con dígitos del 0 al 9 escritos a mano, este conjunto se balancea utilizando la librería train_test_split de Python.

Para simplificar el cálculo, se puede asumir que cada píxel tiene solo dos valores posibles: 0 (blanco) o 1 (negro). Esto se puede lograr aplicando un umbral a las imágenes para convertirlas en binarias. Luego, se puede estimar la probabilidad de cada píxel, dado el dígito, como la frecuencia relativa con la que el píxel está encendido en las imágenes del mismo dígito. Por ejemplo, si el píxel (10,10) está encendido en 300 de las 1000 imágenes del dígito 5, entonces $P(píxel_{10,10} = 1|dígito = 5) = 0.3$.

La probabilidad previa de cada dígito se puede estimar como la proporción de imágenes que tienen ese dígito en el conjunto de datos. Por ejemplo, si hay 7000 imágenes del dígito 5, entonces $P(dígito = 5) = 0.1$.

La probabilidad marginal de los píxeles se puede calcular usando la regla de la suma:

$$P(píxeles) = \sum_{d=0}^{9} P(píxeles|dígito = d)P(dígito = d)$$

Sin embargo, esta probabilidad no afecta a la elección del dígito con mayor probabilidad posterior, ya que es constante para todos los dígitos. Por lo tanto, se puede ignorar y solo comparar los numeradores de las probabilidades posteriores.

Para calcular la probabilidad posterior de cada dígito, dado los píxeles de la imagen, se necesita aplicar el producto de las probabilidades condicionales de cada píxel, dado el dígito, y la probabilidad previa del dígito. Esto se puede hacer usando la regla de la cadena:

$$P(dígito|píxeles) \propto P(píxeles|dígito)P(dígito)$$
$$= P(píxel_{1,1}|dígito)P(píxel_{1,2}|dígito)...P(píxel_{28,28}|dígito)P(dígito)$$

Donde el símbolo $\propto$ significa proporcional a. Esta fórmula implica multiplicar 785 probabilidades para cada dígito, lo que puede causar problemas numéricos por desbordamiento o subdesbordamiento. Para evitar esto, se puede aplicar el logaritmo a ambos lados de la ecuación y usar la propiedad de que el logaritmo del producto es igual a la suma de los logaritmos:

$$\log P(dígito|píxeles) \propto \log P(píxeles|dígito) + \log P(dígito)$$
$$= \log P(píxel_{1,1}|dígito) + \log P(píxel_{1,2}|dígito) + ... + \log P(píxel_{28,28}|dígito) + \log P(dígito)$$

De esta forma, se pueden sumar 785 logaritmos para cada dígito y elegir el que tenga el mayor valor como la predicción.

2 Objetivos:

El siguiente documento tiene como propósito implementar esta metodología de Naive Bayes y en particular desarrollar los siguientes incisos:

1. Elegir un conjunto balanceado de 10.000 imágenes de prueba de dígitos de la base de datos original, con las cuales se diseñará un experimento predictivo que utilice estos datos para evaluar la capacidad de generalización del modelo. Es decir, para un n dado, se especificará cómo se usarán estos datos para medir el desempeño del modelo.
2. Desarrollar un modelo que, dada una imagen, estime un vector con las probabilidades asociadas a cada uno de los dígitos.
3. Reportar el desempeño de generalización del problema de clasificación, usando las siguientes medidas: la precisión, el recall, el F1-score y la matriz de confusión.
4. Desarrollar un modelo que pueda sintetizar un nuevo dígito, es decir, que dado un nuevo dígito, por ejemplo 6, produzca una imagen similar a ese dígito.

3 Desarrollo:

3.1 Especificación:

Se usan las funciones _load_label y _load_img para cargar el conjunto de datos del MNIST. Luego, se usa la función train_test_split de la librería sklearn para dividir el conjunto de datos en un conjunto de entrenamiento y uno de prueba, asegurándose de que ambos estén balanceados. Es decir, que tengan la misma proporción de imágenes para cada dígito. Se usa el parámetro stratify de la función train_test_split para lograr esto. El tamaño del conjunto de prueba es 10000, entonces el conjunto de prueba tiene 10000 imágenes y el conjunto de entrenamiento 50000 imágenes. Obteniendo:

Dimensiones del conjunto de entrenamiento:
(50000, 28, 28, 1)
(50000,)

Dimensiones del conjunto de prueba:
(10000, 28, 28, 1)
(10000,)

3.2 Dada una imagen, estime un vector de probabilidades:

Una vez construidos los conjuntos de entrenamiento y prueba, se procede a construir el modelo de Naive Bayes. Para ello, se necesita calcular las probabilidades a priori de cada clase (P(c), donde c es la clase de cada dígito, p.e. la clase del 0, la del 1, hasta la del 9) y las probabilidades condicionales de cada píxel dado cada clase (P(x—c)). Estas probabilidades se pueden estimar a partir de los datos de entrenamiento usando la regla de Laplace, que consiste en añadir un pequeño valor (por ejemplo, 1) al numerador y al denominador de las frecuencias relativas.

Por lo tanto, la probabilidad a priori de cada clase $\alpha \in \{0, 1, .., 9\}$ (P(c=α)) está dada por:

$$P(C = c) = \begin{pmatrix} 0.09872026 \\ 0.11235753 \\ 0.09930014 \\ 0.10217956 \\ 0.09736053 \\ 0.09036193 \\ 0.09864027 \\ 0.10441912 \\ 0.0975205 \\ 0.09914017 \end{pmatrix}$$

Estas probabilidades siempre serán cercanas a 0.1, que es la probabilidad esperada de cada dígito, dada por:

$$P(C = c) = \frac{\text{Frecuencia del dígito}}{\text{Cantidad de dígitos}} = \frac{1}{10} = 0.1$$

Debido a que son complementarias, su suma es igual a 1.

Para calcular las probabilidades condicionales de cada píxel dado cada clase $(P(x|c))$, se necesita contar el número de veces que cada píxel tiene un valor mayor que cero en las imágenes de cada clase.

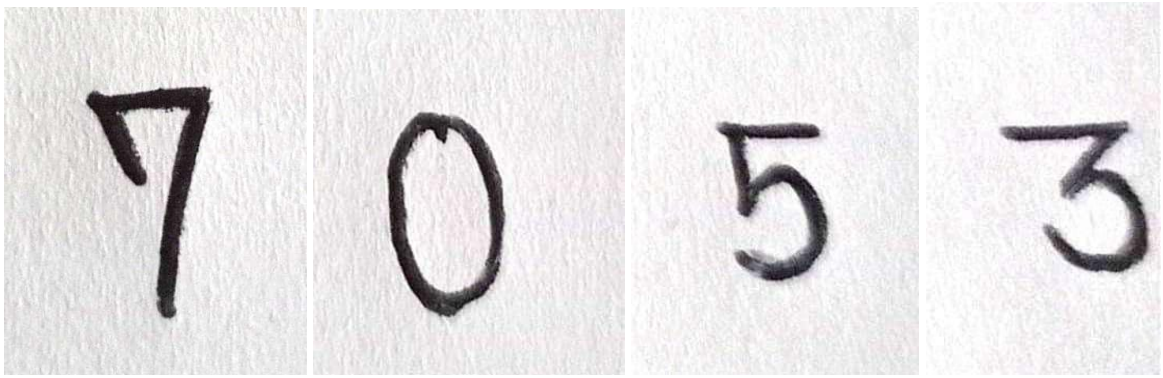
obteniendo la suma de las probabilidades condicionales por clase:

$$\sum P(x|C = c) = \begin{pmatrix} 191.99129202 \\ 85.89715302 \\ 168.5913026 \\ 163.40383487 \\ 141.73367556 \\ 152.74513274 \\ 157.17105797 \\ 131.51158338 \\ 173.57318573 \\ 143.06735229 \end{pmatrix}$$

Notando que la suma de las probabilidades condicionales por clase no tiene que ser 1, porque se definen los pixeles independientes entre sí.

Para construir un modelo que dada una imagen calcule un vector con las probabilidades asociadas a cada uno de los dígitos, usando el teorema de Bayes se definirá una función que tome como entrada una imagen (un vector de 784 píxeles) y devuelva un vector de 10 probabilidades posteriores, una por cada clase.

Se insertan las siguientes 4 imágenes al modelo:



y se obtienen los siguiente vectores:

$$P(C = c|x = 7) = \begin{pmatrix} \mathbf{0.13422114} \\ 0.05986686 \\ 0.11501694 \\ 0.11081846 \\ 0.09207733 \\ 0.09268808 \\ 0.10685148 \\ 0.08729253 \\ 0.11119364 \\ 0.08997352 \end{pmatrix} \qquad P(C = c|x = 0) = \begin{pmatrix} \mathbf{0.12623744} \\ 0.06311433 \\ 0.11396792 \\ 0.112733 \\ 0.09210894 \\ 0.09163634 \\ 0.10408995 \\ 0.08918816 \\ 0.11328372 \\ 0.09364019 \end{pmatrix}$$

$$P(C=c|x=5)=\begin{pmatrix}\mathbf{0.13180546}\\0.06330264\\0.11524962\\0.11225158\\0.09062935\\0.09142604\\0.10092341\\0.08959876\\0.11328776\\0.09152539\end{pmatrix} \qquad P(C=c|x=3)=\begin{pmatrix}\mathbf{0.12574631}\\0.06603828\\0.11218733\\0.11329479\\0.09097346\\0.09385596\\0.10366021\\0.08802603\\0.11342389\\0.09279375\end{pmatrix}$$

De lo anterior, se puede concluir que en todos los casos el dígito con mayor probabilidad es 0, sin embargo no todas las imágenes corresponden a 0's. Luego el modelo de clasificación no es tan preciso.

3.3 Desempeño:

Para reportar el desempeño de generalización del problema de clasificación, se puede usar varias medidas como la precisión, el recall, el F1-score o la matriz de confusión. Estas medidas permiten evaluar qué tan bien el modelo predice la clase correcta para cada imagen. Para calcular estas medidas, se tiene que definir una función que tome como entrada una imagen y devuelva la clase con mayor probabilidad posterior.

Luego, se usará esta función para predecir las clases de todas las imágenes del conjunto de prueba seleccionado (n=1000) y se comparará con las etiquetas reales. Se usarán las funciones accuracy_score, recall_score, f1_score y confusion_matrix de sklearn.metrics para calcular las medidas mencionadas.

Obteniendo:

- Precisión: 0.6526
- Recall: 0.6457322125304122
- F1-score: 0.6236586836657471
- Matriz de confusión:

$$\begin{pmatrix}944&0&1&1&0&0&4&0&36&1\\0&758&4&15&0&0&2&0&343&2\\110&1&663&65&1&0&42&5&105&1\\68&1&12&825&0&0&5&0&107&4\\75&1&16&13&293&0&46&2&223&305\\288&0&3&214&0&0&15&0&360&23\\126&0&10&1&0&0&785&0&64&0\\74&2&2&15&1&0&4&734&161&51\\33&0&1&68&0&0&3&0&869&1\\69&0&9&27&4&0&1&19&208&655\end{pmatrix}$$

Cada una de estas medidad tiene un significado diferente y se calcula a partir de los valores de verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN) y falsos negativos (FN) que se obtienen al comparar las predicciones del modelo con las etiquetas reales.

- La precisión es la proporción de predicciones positivas que son correctas. Se calcula como TP / (TP + FP). Una alta precisión indica que el modelo tiene pocos falsos positivos, es decir, que no clasifica como positivas las instancias que son negativas. En este caso, la precisión es 0.6598, lo que significa que el 65.98% de las predicciones positivas del modelo son correctas.

- El recall es la proporción de instancias positivas reales que son correctamente clasificadas por el modelo. Se calcula como TP / (TP + FN). Un alto recall indica que el modelo tiene pocos falsos negativos, es decir, que no clasifica como negativas las instancias que son positivas. En este caso, el recall es 0.6529, lo que significa que el 65.29% de las instancias positivas reales son correctamente clasificadas por el modelo.

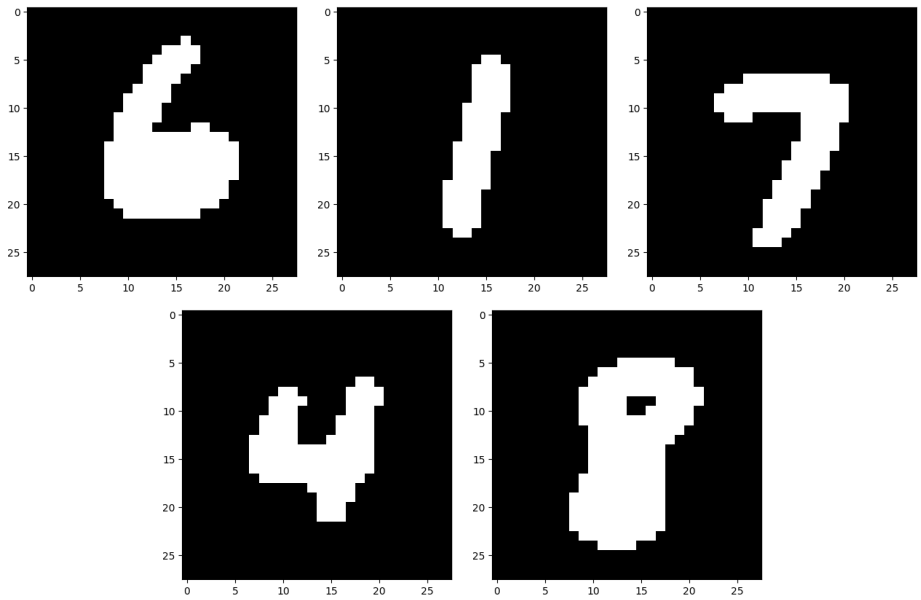
- El F1-score es una medida que combina la precisión y el recall en un solo valor. Se calcula como 2 * (precision * recall) / (precision + recall). Un alto F1-score indica que el modelo tiene un buen equilibrio entre la precisión y el recall, es decir, que tiene pocos falsos positivos y pocos falsos negativos. En este caso, el F1-score es 0.6314, lo que significa que el modelo tiene un rendimiento moderado en términos de precisión y recall.

- La matriz de confusión es una tabla que muestra el número de predicciones correctas e incorrectas para cada clase. Las filas representan las etiquetas reales y las columnas representan las predicciones del modelo. Los elementos de la diagonal principal son los verdaderos positivos, es decir, las instancias que son correctamente clasificadas por el modelo. Los elementos fuera de la diagonal principal son los falsos positivos y los falsos negativos, es decir, las instancias que son incorrectamente clasificadas por el modelo. En este caso, la matriz de confusión tiene 10 filas y 10 columnas, porque hay 10 clases posibles (los dígitos del 0 al 9). Por ejemplo, el elemento (0,0) tiene el valor 954, lo que significa que hay 954 instancias con etiqueta real 0 y predicción del modelo 0. El elemento (1,8) tiene el valor 344, lo que significa que hay 344 instancias con etiqueta real 1 y predicción del modelo 8.

3.4 Sintetizar un nuevo dígito:

Para construir un modelo que permita sintetizar un nuevo dígito, se usa el método de máxima verosimilitud para estimar el valor más probable de cada píxel dado la clase. Este método consiste en encontrar el valor que maximiza la probabilidad condicional de cada píxel dado la clase. Para simplificar el problema, se asume que los valores posibles de cada píxel son solo 0 o 1, es decir, blanco o negro. Entonces, el valor más probable de cada píxel será el que tenga mayor probabilidad condicional. Se define una función que tome como entrada una clase y devuelve una imagen sintetizada para esa clase.

Se prueba con los dígitos 6,1,7,4 y 8, obteniendo:



4 Conclusiones:

1. El modelo de Naive Bayes para esta tarea en particular con este conjunto de datos de entrenamiento, no es capaz de capturar la información total de una imagen nueva y clasificarla correctamente, tan solo clasifica bien un poco más de la midad de la muestra, lo cual es un indicador bastante malo para un modelo de predicción ideal.
2. Según la matriz de confusión, el modelo presenta una confusión bastante amplia inclinada hacia los dígitos 0 y 8, es decir, precide 0 u 8 cuando en realidad es un número diferente.
3. Las imágenes generadas tienen mucho relleno en su estructura, por ejemplo, las imagenes generadas para los dígitos 6 y 8, tal vez sea esto lo que influye en la confusión del modelo.

5 Webgrafía:

- https://addi.ehu.es/bitstream/handle/10810/19053/TFM_OARV.pdf?sequence=1
- https://www.kaggle.com/datasets/hojjatk/mnist-dataset?resource=downloadselect=train-labels-idx1-ubyte
- https://medium.com/datos-y-ciencia/algoritmos-naive-bayes-fudamentos-e-implementaci%C3%B3n-4bcb24b307f
- https://github.com/AprendizajeProfundo/Diplomado/blob/master/Talleres/Cuadernos/DataLoaders-MNIST.ipynb
- https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1007007