Predicción usando modelo de regresión logística optimizado con PCA

Frado García-Palacios[1]

[1] *Tecnologico de Monterrey, IDM, Jalisco, Mexico*

Fecha de publicación: 24/11/2023

Resumen— Esta estudio propone la mejora de modelos de aprendizaje automático supervisado, en concreto, modelos de regresión logística con ayuda de la previa aplicación de PCA a las bases de datos a analizar. Se usaron varias bases de datos para corroborar esta hipótesis y poder visualizar los resultados de las predicciones hechas con estos modelos mejorados. Se encontró que en efecto, la aplicación previa de PCA a las bases de datos, mejora no solo los tiempos de ejecución del código en Python, sino que también mejora ligeramente la exactitud de los modelos.

Palabras clave— PCA, regresión logística, optimización, base de datos.

I. Introducción

La predicción de sucesos futuros es una de las características dentro del pensamiento mágico que ha acompañado a la humanidad desde milenios atrás. Tan solo basta investigar algunas de las obras más importantes de la antigua cultura griega para percatarse de la importancia de los oráculos; personas con la capacidad de recibir mensajes de los dioses acerca del futuro de la humanidad [1].

Si bien, estas tradiciones han cambiado bastante con el pasar de los años, la humanidad continúa en su búsqueda por la verdad de forma que ha logrado crear métodos más sofisticados para poder predecir estos sucesos futuros con un nivel de certeza o confianza más elevado.

Uno de estos métodos es la regresión logística, el cual es un modelo capaz de clasificar conjuntos de observaciones con variables independientes discretas o continuas en una variable discreta dependiente que puede ser binomial o multinomial [2] [3]. El método por sí solo funciona bien para predecir resultados en campos como la medicina, economía o ciencias naturales y si se toma en cuenta que es automatizable con ayuda de herramientas sencillas de programación, se obtiene un método muy sencillo de aplicar e interpretar.

Como muchos modelos de aprendizaje automático supervisado, la regresión logística tiene sus limitaciones cuando se le presentan bases de datos con cantidades grandes de variables independientes y aquí es donde entra el uso del PCA para reducir la dimensionalidad [4] de las bases de datos y, por tanto, reducir el número de variables independientes a analizar.

El objetivo de este estudio es determinar la mejora en los tiempos de procesado al emplear regresión logística con bases de datos donde se ha aplica PCA para reducir su dimensionalidad y comparar los resultados en modelos donde no se ha aplicado PCA o se ha utilizado un número distinto de vectores arrojados por este.

Contact data: Frado García, fradog2003@gmail.com

II. Metodología

La metodología a seguir para poder determinar la mejora en el modelo de regresión logística con PCA se puede apreciar el diagrama de flujo (ver Fig.1).

a. Ambiente de trabajo

Se usó el ambiente de `Jupiter Notebook` para programar los modelos usando el lenguaje de programación `Python`. A su vez, se emplearán las librerías, `pandas` y `numpy` para el manejo y análisis de las bases de datos; `matplotlib` para generar gráficos de los resultados y `sklearn` para realizar los modelos de regresión logística y aplicar PCA a las bases de datos.

b. Bases de datos

Con el propósito de explorar los resultados de la regresión logística en distintos campos, se extrajeron tres bases de datos con distinta cantidad de variables independientes y variables dependientes binomiales y multinomiales. Se uso, la base datos `Wine Quality` [5] que determina la calidad de distintas muestras de vino rojo y da información sobre su composición; la base de datos `Fashion MNIST` [6] que recopila imagenes (28×28) de muestras de 10 tipos de prendas de ropa e información sobre los pixeles que conforman las imagenes; y la base de datos `Heart Disease Prediction` [7] que recopila variables clinicas de distintas personas y si presentan alguna enfermedad cardiovascular.

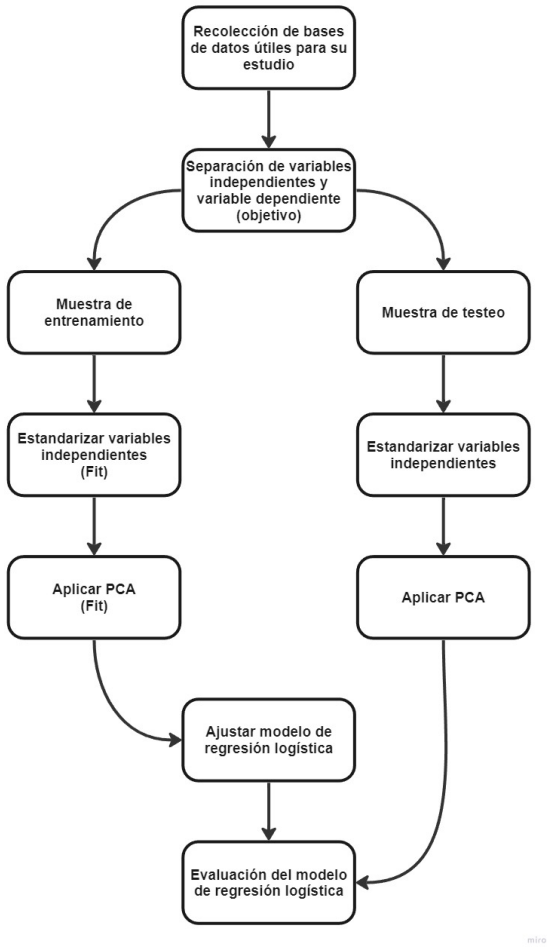

Fig. 1: Metodología a seguir para completar el objetivo

c. Proceso de modelación

Para una mayor practicidad, solo se mostrará el proceso de modelación con regresión logística empleado en la base de datos `Fashion MNIST` [6].

Se cargan las librerías previamente mencionadas.

```
#librerias basicas para manejo de datos y creacion de graficos
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

#Libreria para estandarizar datos
from sklearn.preprocessing import StandardScaler

#Libreria para aplicar PCA y crear los nuevos vectores
from sklearn.decomposition import PCA

#Libreria para crear el modelo regresion logistica y predecir
from sklearn.linear_model import LogisticRegression

#Libreria para crear matriz de confusion
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
```

Listing 1: Carga de librerías

Después se cargaron las bases de datos de entrenamiento y prueba.

```
#Base de datos de entrenamiento
df_f_mnist_train = pd.read_csv('fashion-mnist_train.csv')

#Base de datos para prueba
df_f_mnist_test = pd.read_csv('fashion-mnist_test.csv')
```

Listing 2: Carga de bases de datos

Se extrajeron los valores de las variables independientes y de la dependiente.

```
#Separacion para base de datos de entrenamiento
df_f_mnist_train_X = df_f_mnist_train.iloc[:,1:].values
df_f_mnist_train_Y = df_f_mnist_train.iloc[:,0].values

#Separacion para base de datos de prueba
df_f_mnist_test_X = df_f_mnist_test.iloc[:,1:].values
df_f_mnist_test_Y = df_f_mnist_test.iloc[:,0].values
```

Listing 3: Extracción de variables

Para poder aplicar PCA a la base de datos, se requiere estandarizar cada una de las observaciones de forma que a estas se les reste el promedio y sean divididas entre la desviación estandar de la variable en la que se encuentran. Esto es necesario para equilibrar las varianzas de cada variable a una varianza igual a 1 y que esta no afecte el desempeño del PCA ya que, de no realizarse esta estandarización, las variables con mayor varianza predominarían en el análisis, dando a entender que son más importantes para la predicción de la variable dependiente aunque esto no sea verdad.

$$z = \frac{x_n - \bar{X}}{S} \tag{1}$$

Es importante destacar que todas las observaciones de las bases de datos de entrenamiento y prueba serán estandarizadas, sin embargo, la estandarización solo se ajustará a la base de datos de entrenamiento de forma que no tomen en cuenta a la base de datos de prueba para calcular la media y desviación estandar. Esto es importante debido a que no debe existir ninguna relación entre los parámetros del modelo y las observaciones de prueba para que podamos al final probar el modelo y determinar su fiabilidad.

```
#Se llama a la funcion para estandarizar
scal = StandardScaler()

#Se ajusta la estandarizacion solo a los datos de entrenamiento
scal.fit(df_f_mnist_train_X)

#Se aplica la estandarizacion a ambas matrices con parametros de la matriz de entrenamiento
train_X_st = scal.transform(df_f_mnist_train_X)
test_X_st = scal.transform(df_f_mnist_test_X)
```

Listing 4: Estandarización

Ahora que los valores de ambas bases de datos ya estan estandarizados, podemos proceder a aplicar el análisis PCA para disminuir la dimensionalidad de la base de datos. Los vectores obtenidos no poseerán de significado intrinseco pero cumplirán con la propiedad de que serán ortogonales entre sí y, por tanto, no estarán correlacionados.

De nuevo, el análisis PCA solo se ajustará a la base de datos de entrenamiento de forma que pueda aplicarse a ambas bases de datos una vez ajustado el PCA.

Se ha programado la extracción de los vectores propios de la matriz de covarianzas para que solo se obtengan los primeros vectores cuya suma de porcentajes de varianza de un 95%. Se redujo la cantidad de variables independientes de 784 a 256; la dimensionalidad de las bases de datos se redujo a aproximadamente $\frac{1}{3}$.

```
#Se llama a la funcion para aplicar PCA y crear los vectores
pca = PCA(0.95) #Se requieren los primeros vectores cuya suma de porcentaje de varianza sea el 95%

#Se ajusta el PCA solo para la matriz de entrenamiento
pca.fit(train_X_st)

#Se aplica PCA a ambas matrices y se obtienen los nuevos vectores ortogonales
train_X_pca = pca.transform(train_X_st)
test_X_pca = pca.transform(test_X_st)
```

Listing 5: PCA

Ya se ha reducido la dimensionalidad de ambas bases de datos de entrenamiento y prueba. Ahora podemos entrenar el modelo de regresión logística con la matriz de entrenamiento de 256 vectores propios y el vector de entrenamiento con la variable de respuesta o dependiente.

Se entrenó el modelo usando el algoritmo de optimización `lbfgs` (Limited-memory BFGS) ya que es de los más rápidos para modelos donde la variable de respuesta es multinomial.

```
#Se llama a la funcion de regresion logistica con el algoritmo lbfgs
logisticRegr = LogisticRegression(solver = 'lbfgs')

#Se aplica regresion logistica a las variables independientes y dependiente de entrenamiento
logisticRegr.fit(train_X_pca, df_f_mnist_train_Y)
```

Listing 6: Regresión logística

Para este punto, el modelo de regresión logística ya está entrenado para predecir uno de los 10 tipos de prenda para cualquier observación de pixeles que le demos.

Este mismo proceso es replicable para las otras dos bases de datos empledas con la única diferencia en que estas debemos dividirlas en muestras de entrenamiento y prueba. Este procedimiento es muy sencillo con la ayuda de la función `sklearn.model_selection.train_test_split`; solo es necesario indicar a la función que porcentaje de observaciones deseas utilizar como muestra de prueba.

```
train_X, test_X, train_Y, test_Y = train_test_split(X_variables, Y_target, test_size=0.15, random_state=0)
```

Listing 7: Función para separar base de datos

III. Análisis de resultados

Para cuantificar los resultados del modelo de predicción ajustado a la base de datos `Fashion MNIST` [6] que se realizó previamente, se usará el estadístico de exactitud que compara cada uno de los valores predichos con el vector de respuesta

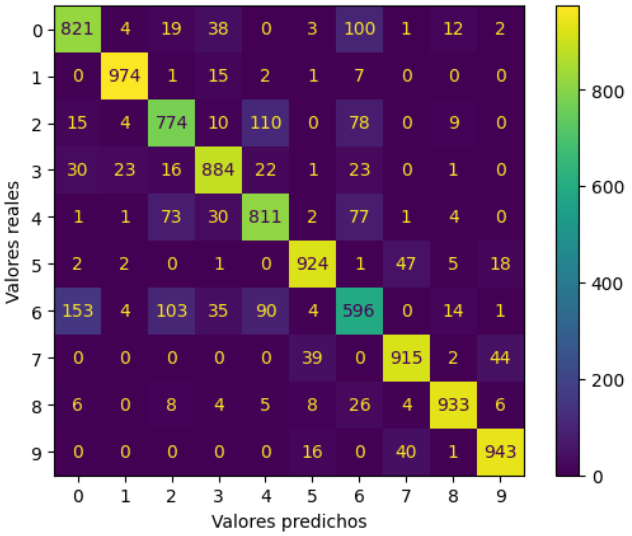

Fig. 2: Matriz de confusión del modelo asociado a la base de datos Fashion MNIST

que contiene los valores verdaderos. Se obtuvo un 85.75% de exactitud; esto quiere decir que aproximadamente, un 85.75% de las observaciones que le demos al modelo serán predicciones correctas. Este valor no es tan alto como para aprobar el modelo, sin embargo, considerando que algunas de las prendas que tuvo que distinguir eran muy parecidas, podemos tomo ese 85.75% de exactitud como un acierto con a base a los resultados esperados.

Para ver cuales fueron las prendas que más confundió a la hora de probar el modelo, podemos apoyarnos de una matriz de confusión (ver Fig. 2).

Cada número representa una prenda de ropa:

- 0: Camiseta/top
- 1: Pantalón
- 2: Suéter
- 3: Vestido
- 4: Abrigo
- 5: Sandalia
- 6: Camisa
- 7: Zapatos deportivos
- 8: Bolsa
- 9: Botín

Se puede observar que la prenda que mayor dificultad tuvo para predecir fue la camisa que la confundió primordialmente con la camiseta/top, suéter y el abrigo. Este resultado es bastante lógico debido a las similitudes en forma de estas prendas. Si la base de datos almacenara otro tipo de variables independientes como la textura o grosor de la tela, o bien, variables más subjetivas como la comodidad y la estética; se podría mejorar el análisis para que el algoritmo no confundiera estas prendas.

Dimensión	Exactitud	Tiempo de ejecución
2	0.5059	2 seg.
256	0.8575	14 seg.
784	0.8544	50 seg.

TABLE 1: TABLA COMPARATIVA ENTRE LA DIMENSIONALIDAD

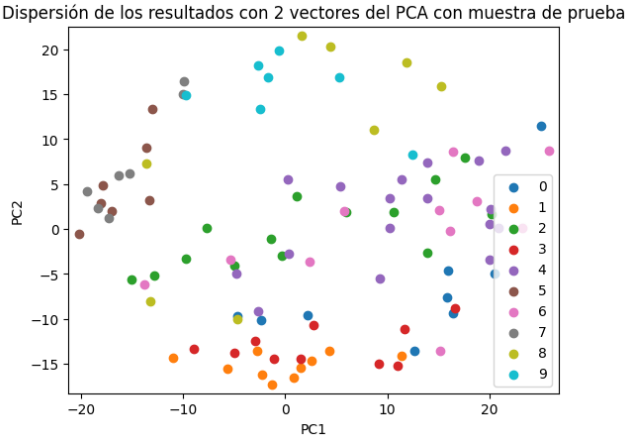

Fig. 3: Gráfica de dispersión con 2 vectores PCA de la base de datos Fashion MNIST

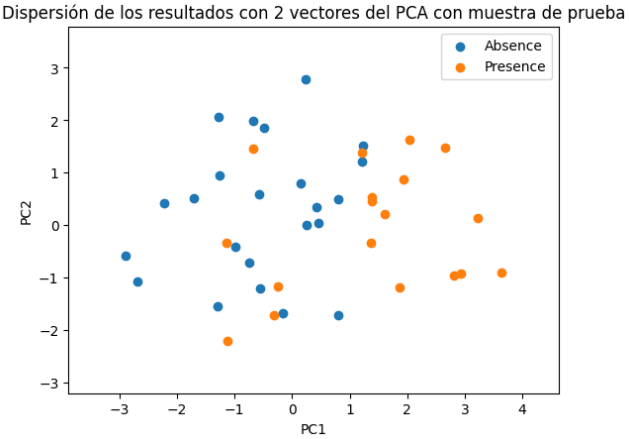

Fig. 4: Gráfica de dispersión con 2 vectores PCA de la base de datos Heart Disease Prediction

a. Mejora en los tiempos de ejecución

Uno de los objetivos de aplicar PCA a las bases de datos para posteriormente generar modelos de predicción; es justamente mejorar los tiempos de ejecución del código (ver Tab. 1).

Se aprecia que con una menor dimensionalidad se obtuvo una mejor exactitud y se logró reducir el tiempo en un 72%

b. Visualicación de 2 vectores PCA

Agrupación de los resultados con 2 vectores de PCA en la base de datos `Fashion MNIST` [6] (ver Fig. 3)

Agrupación de los resultados con 2 vectores de PCA en la base de datos `Heart Disease Prediction` [7] (ver Fig. 4)

IV. CONCLUSIONES

Los resultados obtenidos fueron los esperados ya que usando PCA se logró disminuir los tiempos de ejecución e incluso mejor la exactitud de los modelos. Se da por hecho que los puntajes de exactitud no tan altos se deben a la falta de información por parte de las bases de datos al describir los objetos a predecir y solo centrarse en unos cuantos criterios.

La regresión logística con ayuda del PCA, probó ser una buena herramienta para predecir eventos de la vida cotidiana.

V. AGRADECIMIENTOS

Se agradece al repositorio UC Irvine Machine Learning Repository, Zalando y Kagle por la distribución de las bases de datos empleadas en este estudio.

A. APÉNDICE

a. Material suplementario

Los archivos de las bases de datos empleadas y el código se puede encontrar en Repositorio en GitHub o puedes acceder al siguiente url: `https://github.com/Fradeiro2003/PCA_y_regresion_logistica`

REFERENCES

[1] T. E. of Encyclopaedia, "Delphic oracle," *Encyclopedia Britannica*, 2023. [Online]. Available: https://www.britannica.com/topic/Delphic-oracle

[2] A. Subasi, "Chapter 3 - machine learning techniques," in *Practical Machine Learning for Data Analysis Using Python*, A. Subasi, Ed. Academic Press, 2020, pp. 91–202. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128213797000035

[3] K. Pereira Teodoro da Silva, A. Kalbusch, and E. Henning, "Detection of unauthorized consumption in water supply systems: A case study using logistic regression," *Utilities Policy*, vol. 84, p. 101647, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957178723001595

[4] C. Zhou, L. Wang, Q. Zhang, and X. Wei, "Face recognition based on pca and logistic regression analysis," *Optik*, vol. 125, no. 20, pp. 5916–5919, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030402614008511

[5] P. Cortez, "Wine quality," *UC Irvine Machine Learning Repository*, 2009. [Online]. Available: https://doi.org/10.24432/C56S3T

[6] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. [Online]. Available: https://github.com/zalandoresearch/fashion-mnist

[7] R. H. MD, "Heart disease prediction," *data.world*, 2019. [Online]. Available: https://doi.org/10.24432/C52P4X