ISSN: 2007-4948

https://repository.uaeh.edu.mx/revistas/index.php/xikua/issue/archive

XIKUA Boletín Científico de la Escuela Superior de Tlahuelilpan
13º Congreso Internacional de Computación
Inteligencia artificial: Presente y futuro
Red Iberoamericana de Computación
Vol. 12, Número Especial (2024) 71-80

Regresión Logística Técnica de Machine Learning para predicciones académicas

Logistic Regression Machine Learning Technique for academic predictions

Nelson Becerra Correa[a], *Miguel Angel Leguizamón Páez*[b]

Abstract:

In this article, we use logistic regression as a machine learning technique to predict the probability that a first-semester student at the Francisco José de Caldas District University will fail or pass the calculus subject, and with what percentage.

To do this, make use of the student database available in the data repository of the FJC district university, see [1].

Initially, we used logistic regression using the student's entry age and the grade obtained in previous semesters, to predict the percentage with which a new student would obtain a passing grade according to the standards of the district university, which is three point zero (3.0).

As a second term we use multiple logistic regression, with it we use more than two input variables and the grade obtained by the students in the previous semester. With this we predict the grade of the student with a passing percentage.

Firstly, using simple logistic regression, we predict the probability that a student with a given score on the state tests (ICFES) will fail a first semester course at the FJC District University of Bogotá.

For this we have used Python and the keras and tensorflow libraries. To evaluate the efficiency of our model, we have analyzed the data using: The mean square error, Mean Squere error´ (MSE), The root of the MSE, Root Mean Square Error (RMSE) and the coefficient of determination (R2).´ We have also evaluated our model using the square root of the mean square error, RMSE: Root Mean Square' Error, which measures how well a regression line fits the data points.

For the management of the database, we worked on CVS files and they were manipulated using NumPy and Pandas libraries for this purpose supplied by Python.

Keywords:

Simple Logistic Regression, Multiple Logistic Regression, Mean Squere Error (MSE), Coefficient of Determination (R2), Gradient Descent, Student

Resumen:

[a] Universidad Distrital Francisco José de Caldas, Facultad Tecnológica, Bogotá – Cundinamarca, Colombia. https://orcid.org/0009-0000-5522-8481, nrbecerrac@udistrital.edu.co
[b] Autor de correspondencia, Universidad Distrital Francisco José de Caldas, Facultad Tecnológica, Bogotá – Cundinamarca, Colombia. https://orcid.org/0000-0003-0457-0126, maleguizamonp@udistrital.edu.co

Fecha de recepción: 13/04/2024, Fecha de aceptación: 22/05/2024, Fecha de publicación: 01/07/2024

DOI: https://doi.org/10.29057/xikua.v12iEspecial.12746

En este artículo, empleamos la regresión logística como técnica de machine learning para predecir cual es la probabilidad que un estudiante de primer semestre de la Universidad Distrital Francisco José De Caldas repruebe o aprueba la materia de cálculo y con qué porcentaje.

Para ello hacer uso de la base de datos de estudiantes disponible en el repositorio de datos de la universidad distrital FJC ver [1].

Inicialmente utilizamos regresión logística utilizando la edad de ingreso del estudiante y la nota obtenida en semestres anteriores, para predecir el porcentaje con el cual un nuevo estudiante obtendría una nota aprobatoria según las normas de la universidad distrital la cual es tres puntos cero (3.0).

Como segundo término utilizamos regresión logística múltiple, con ella utilizamos más de dos variables de entrada y la nota obtenida por los estudiantes en el semestre anterior. Con ello predecimos la nota del estudiante con un porcentaje de aprobación.

En primer término, mediante regresión logística simple, predecimos, la probabilidad de que un estudiante con determinado puntaje en las pruebas de estado (ICFES) repruebe una materia de primer semestre en la Universidad Distrital FJC de Bogotá. Para ello hemos utilizado Python y las librerías keras y tensorflow. Para evaluar la eficiencia de nuestro modelo, hemos analizado los datos mediante: El error cuadrático medio, Mean Squere error´ (MSE), La raíz del MSE, Root Mean Square

Error (RMSE) y el coeficiente de determinación (R2). también hemos evaluado nuestro modelo mediante, la raíz cuadrada del error cuadrático medio, RMSE: Root Mean Square´ Error que nos mide que tan bien se ajusta una línea de regresión´ a los puntos de datos.

Para el manejo de la base de datos se trabajó sobre archivos CSV y fueron manipulados mediante NumPy y Pandas librerías para tal fin suministradas por Python.

Palabras Clave:

Regresión logística simple, Regresión logística múltiple, Mean Squere error (MSE), Coeficiente de determinación (R2), Descenso del gradiente, Estudiante.

Introducción

Este documento está estructurado de la siguiente manera: En la primera sección se definen los conceptos básicos sobre aprendizaje, se definen los tipos de aprendizaje y los conceptos básicos del problema. La segunda sección describe en detalle la regresión logística como método de aprendizaje y sus tipos de regresión.

La fase tres se da una explicación sobre la forma y la estructura necesaria para la adaptación del método de regresión logística al problema reprobación de estudiantes en la Universidad Distrital FJC

Igualmente se describe en detalle la estructura del método de regresión lineal múltiple y se describen las variables utilizadas para modelar el problema

En la cuarta fase, se detalla el diseño, del modelo de regresión en el lenguaje de programación Phyton y las librerías utilizadas lo mismo que la base de datos utilizada.

La quinta fase muestra el resultado del modelo probado con la base de datos de estudiantes de la Universidad Distrital FJC de año 2020-2021.

La fase sexta se proponen los trabajos futuros a realizar y como último aspecto se presentan las conclusiones del artículo.

Objetivo general

Modelar como influyen ciertas características como sexo, edad, puntaje del icfes, lugar de procedencia en el rendimiento académico en la materia de cálculo de la Universidad Distrital, para los estudiantes de primer semestre.

Objetivos específicos

1. Modelar e implementar el algoritmo de regresión logística como predicción académica en la materia de cálculo de la Universidad Distrital Francisco José de Caldas
2. Obtener y depurar la base de datos histórica de estudiantes de la Universidad Distrital Francisco José De Caldas

3. Determinar el modelo mejor ajustado el cual sea razonable que permita describir la relación entre la variable dependiente y las variables regresaras o independiente.
4. Utilizar Python como lenguaje de programación.

Aprendizaje

En la literatura actual se habla de fundamentalmente de tres tipos de aprendizaje estos son: Aprendizaje supervisado, no supervisado y aprendizaje por refuerzo.

A continuación, describiremos estos tres tipos de aprendizaje [7] y para los destinos métodos de machine learning si se desea profundizar remitimos a [8,9,10,11,12,13,14]

Dicho tema ya fue explicado por los autores en el artículo, Regresión lineal como herramienta de machine learning para análisis de datos por tal razón remitimos a https://comunidad.udistrital.edu.co/cicom2021/memoriasdelevento/

Academia

La Universidad Distrital Francisco José de Caldas de Bogotá ha implementado políticas de reforzamiento de deficiencias en los alumnos con el fin de evitar la deserción académica la cual tiene varias causas entre ella la perdida de materias en primer semestre debido a la deficiencia académica con la cual vienen los estudiantes del ciclo de educación media. Como mecanismo para contrarrestar esto fenómeno se creó en la universidad distrital un programa para reforzar esta problemática.

Para mayor detalle remitimos a [2].

Como aporte a esta problemática nosotros planteamos utilizar técnicas de Machine Learning para la detección temprana de estudiante con déficit académicos debido a la formación del ciclo de educación media.

Regresión logística

La regresión logística, es una de las técnicas estadísticas utilizadas en machine learning como paradigma del aprendizaje supervisado.

Cuya finalidad es hacer clasificaciones. Con ello podemos crear un modelo que tome una variable y determine la posibilidad de la ocurrencia de un fenómeno.

Amanera de ejemplo dada una distancia en metros y un blanco para el tiro en arco. Determinar la probabilidad que acierte o no el blanco.

Luego de ejecutar algunos tiros con el arco vemos que cada vez que nos alejamos del blanco acertamos menos. Hablando en termino de probabilidad nuestro modelo de regresión logística debería predecir que cuando la distancia es grande tendremos una probabilidad baja.

Las tareas de regresión se preocupan en estimar la relación entre alguna variable de entrada con una variable de salida continua. Podemos partir de la regresión lineal, que es un método estadístico que se utiliza cuando queremos obtener el valor de una variable en función de otra u otras variables.

Por ejemplo, si tenemos un conjunto de datos con 1000 observaciones y 2 variables, altura y peso. Necesitamos predecir el peso dada la altura. La ecuación sería el de Regresión Lineal simple ya que solamente cuenta con una variable independiente y se puede escribir de la siguiente forma:

$$Y = ax + b \quad (1)$$

Y= ax + b En donde: Y es el peso, X la altura y a, b son los coeficientes a ser calculados.

En la regresión lineal la salida Y esta en las mismas unidades que la variable objetivo, a diferencia en la regresión logística la salida Y esta en cuota logarítmica.

Entendemos por cuota la probabilidad de que ocurra un evento.

Cuota = p(evento) / {1-p(evento)}

Para nuestro ejemplo, si hemos disparado 100 flechas e hicimos 60 aciertos. La probabilidad de acertar si hacemos un disparo
será de 60%
Cuota = 0.60 / (1-0.60) =
Si reescribimos nuestra ecuación de regresión logística como

Y = $B0$+ $B1$*Distancia

Donde Y es el log (cuota de acertar).

Haciendo huso de la función sigmoide, podemos reescribir la probabilidad de acertar como:

Probabilidad de acertar 1/[1+$e-y$]

Siendo Bo y B1 los coeficientes a estimar en la ecuación de regresión logística, para mayor detalle remitimos a ver [2].

En otras palabras, nos indica cual es la relación entre la variable de entrada x y la variable de salida y. b es el termino independiente que indica en qué punto la recta corta a la variable dependiente, es decir, indica lo rápido que crece o decrece la recta. Este será el caso de la estructura de regresión logística simple. En donde tenemos únicamente una variable de entrada.

La regresión logística es adecuada cuando se necesita modelizar una variable respuesta binaria, del tipo aprobación o perdida de materia, y permite el uso conjunto de covariables de tipo categórico y continuo, proporcionando interpretación biológica a sus parámetros [9].

En ella se suele simbolizar con Y a la variable respuesta, del tipo presencia (Y = 1) o ausencia (Y = 0) de enfermedad y con (X) a la siguiente probabilidad: P (Y = 1/X), donde X es un vector de p covariables.

El modelo de regresión logística múltiple esta dado por:

g(X)=β0 +β1X1 +...+βpXp,

donde:

Y, es la variable dependiente y X1, X2, . . ., son las variables independientes o explicativas. De otro lado la ecuación de la función sigmoide es

$$Y = \frac{1}{1 + e^{-x}}$$

luego si aplicamos la función sigmoide en la regresión lineal tendremos:

$$\mathrm{P} = \frac{1}{1 + e^{-(a_1 x_1, a_2 x_2, a_3 x_3, \ldots a_4 x_4, + b)}}$$

Para poder determinar los coeficientes que mejor se ajustan a nuestro modelo podemos utilizar 2 métodos el método de mínimos cuadrados y el método de máxima verosimilitud.
Temas estos explicados en cicom2022

El método de máxima verosimilitud se puede aplicar a métodos lineales como no lineales. En el caso de la regresión logística el objetivo como se menciona es encontrar los parámetros B0, B1,… que maximice la función logaritmicall(0) denominada función de verosimilitud.

Donde L (0) es la probabilidad de que se produzcan los datos como son observados.

Función de costo

La función de costo trata de medir que tan equivocados estamos. En nuestro ejemplo cuantos disparo dieron en el blanco. Pero en el caso que no haya acierto implica que deberá haber un costo si el error fue pequeño el costo es pequeño si no fue muy grande el costo deberá ser medio o si el error fue muy grande el costo deberá ser alto.

En el caso de la regresión lineal basta con encontrar la diferencia entre el resultado real y el predicho y elevarla al cuadrado. Sin embargo, en la regresión logística el valor de la función objetivo varía entre 0 y 1.

De vuelta al ejemplo, si hice un disparo a una distancia de 1 metro, y he acertado. Mi modelo ha predicho una probabilidad de 0.95 de precisión, sin embargo, yo he acertado lo cual debería arrojar el modelo 100%. Como no lo hizo el modelo se equivocó un poco en consecuencia lo penalizamos poco, 0,0513 que será el logaritmo de 0.95. Sin embargo, el error de clasificación si lo hemos definido que si el valor por defecto era del 50% el modelo habría predico correctamente. Como el modelo no estaba seguro 100% se penaliza por su incertidumbre.

De esta manera si el resultado real es 1, entonces el costo es /log (probabilidad predicha). Si el resultado real es 0, entonces el costo es /Logaritmo (1-probabilidad predicha).

El costo total lo obtendríamos sumado las probabilidades individuales, mediante el anterior procedimiento. Luego lo minimizaríamos utilizando el descenso del gradiente descrito en bibliografía cicom2022 y CICOM 2021 ver [15]. De esta manera obtendríamos $B0+ B1$ que minimizarían la función de costo.

$B1$ mide que impacto tiene la distancia en la precisión del disparo y $B0$ la intersección que es la predicción de cuota logarítmica del modelo.

Como Y esta dada en probabilidades, utilizamos la función sigmoide para determinar la probabilidad de acertar en un disparo.

Modelo Logit

Este apartado toma como base ver [15], el modelo logit calcula el logaritmo de la razón de monomios y la calculamos de la siguiente manera (P/1-P), para nuestro ejemplo si la probabilidad de acertar la sesta es de 60%, se interpreta como que existe una posibilidad de 6 a 4 de acertar la canasta.

Nosotros hemos seguido el siguiente método de logit.

1. Luego de hacer la regresión Li = Ln(p/1-p)
2. Luego utilizamos la función inversa del logaritmo neperiano que es la exponencial (P/1-P) = e* el resultado del punto 1. Donde P/1-P = e*, p= e* - p - e*

En consecuencia, P se puede estresar de la siguiente manera en términos de la variable independiente

$$\boldsymbol{P}=\mathbf{1}/(1+e-(**x-**))$$

Base de datos de estudiantes

Para nuestro trabajo, utilizamos la base de datos de estudiantes de la Universidad Distrital Francisco José de Caldas la cual contiene los siguientes datos.

Diseño de la solución

Nuestro prototipo, utiliza como lenguaje de programación Phyton y las librerías tflearn por su simplicidad en la implementación.
Se ha elegido el lenguaje Python por tener licencia de código abierto denominada Python Software Foundation License [12]. Además, por hacer sencillas las implementaciones de regresión logística ya que posee librerías desarrolladas para la implementación de aprendizaje profundo (deep learning).

Dentro de estas librerías de Python tenemos Scikit Lear, que es una de las librerías más utilizadas en Machine Learning. ScikitLearn facilita una función LinearRegression para implementar regresión lineal. sklearn la cual nos permite importar linearmodel, evitando de esta manera hacer los cálculos.
Para mayor detalle remitimos a: [3]

En este apartado utilizaremos el criterio de mínimos cuadrados y el escalado de datos.

Implementación mediante mínimos cuadrados
1) Preparación de datos: Preparar los datos fue el primer paso realizado. Para ello esto se importó la base de datos de:

ConfirmedCasesraw=pd:readcsv(https://raw:githubus
ercontent
:com/CSSEGISandData

Escalado de los datos

Para aplicar un escalado a los datos tenemos 2 enfoques; escalado y normalización.
En el escalado los valores son transformados en un rango de valores a,b típicamente en 0,1 o -1,1 .
En la normalización las características de las variables son transformados de tal manera que estas comparten la misma media y desviación media.
Algo importante es indicar que el escalador deberá entrenarse solo con datos de entrenamiento, y aplicarse a los datos de entrenamiento validación y prueba.
En las cuales contiene el número de estudiante con sus respectivas notas.

```
from sklearn impor linear_model
x_entrenamiento =
    VariablesIndependientesEntrenamiento

y_entrenamiento =
    VariablesDependientesEntrenamiento
x_prueba =
    VariablesIndependeientesPrueba
y_prueba =
    variablesDependientesPrueba
```

Recuperamos el algoritmo a utilizar:

```
algoritmo = linear_model.
    LinearRegresion()
algoritmo.fit(x_entrenamiento,
    y_entrenamiento)
     algoritmo.predict(x_prueba)
```

Con las líneas anteriores importamos las librerías, creamos parámetros de entrenamiento y llamamos al método con los datos a entrenar.

Como hemos mencionado antes, la regresión lineal, se basa en encontrar la intersección y la pendiente de la recta o el plano que mejor se ajuste a los datos. Para conocer el valor de la pendiente solamente tenemos que utilizar el comando COEF, una vez ejecutemos este comando nos retorna el valor de a. Ejemplo a = algoritmo:coef
Para conocer la intersección de la recta con el eje Y basta llamar a intercept, en consecuencia b = algoritmo:intercept. Esto para la regresión simple.

Implementación:

Primer paso: Importamos librerias

from sklearn import datasets import pandas as pd import numpy as np Preparamos los datos

Segundo paso: Preparamos los datos utlizando la base de
datos de estudiantes

########## PREPARAR LA DATA ##########

dataset = pd.read_csv('EstCalculoPrueva.csv', sep=",")
print(dataset)

Tercer paso: entendiendo los datos

########## ENTENDIMIENTO DE LA DATA
#Verifico la información contenida en el
dataset print('Información en el dataset:')
print(dataset.keys())
print()

Cuarto paso: Verificamos los datos del dataset

#Verifico las características del dataset
print('Características del dataset:')

print(dataset.CODIGO)

#Seleccionamos todas las columnas
print('Características del DATA.....:')

Quinto paso:
Comparamos los efectos que tiene el estrato en la nota del
estudiante

X = dataset.ESTRATO
X = np.array(X).reshape(-1,1)

Sexto paso: Definimos los datos correspondientes a las etiquetas.

y = dataset.Nota
print("Caracteristicas de la variable X.(Estrato)....:")
print(X)
print("Características de la variable y.(Nota).....:")
print(y)

Séptimo paso: Implementamos la regresión logística

from sklearn.model_selection import train_test_split

Separamos los datos de "train" en entrenamiento y prueba para probar los algoritmos

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

Se escalan todos los datos from
sklearn.preprocessing import
StandardScaler escalar =
StandardScaler() X_train =
escalar.fit_transform(X_train)
X_test = escalar.transform(X_test)

Definimos el algoritmo a utilizer

from sklearn.linear_model import
LogisticRegression algoritmo =
LogisticRegression() #Entreno el modelo
algoritmo.fit(X_train, y_train)

Octavo paso: Realizamos una predicción
y_pred = algoritmo.predict(X_test)

Noveno paso : evaluamos los resultados

Verificamos la matriz de Confusión

from sklearn.metrics import confusion_matrix
matriz = confusion_matrix(y_test, y_pred)
print('Matriz de Confusión:')
print(matriz)

Calculamos la precisión del modelo

from sklearn.metrics import precision_score precision = precision_score(y_test, y_pred,average=None)
print('Precisión del modelo:')
print(precision)

Calculamos la exactitud del modelo
from sklearn.metrics import accuracy_score exactitud = accuracy_score(y_test, y_pred)
print('Exactitud del modelo:')
print(exactitud)

Calculamos la sensibilidad del modelo

from sklearn.metrics import recall_score sensibilidad = recall_score(y_test, y_pred,average=None)
print('Sensibilidad del modelo:')
print(sensibilidad)

Calculamos el Puntaje F1 del modelo from sklearn.metrics import f1_score puntajef1 = f1_score(y_test, y_pred,average=None)
print('Puntaje F1 del modelo:')
print(puntajef1)

Calculamos la curva ROC - AUC del modelo from sklearn.metrics import roc_auc_score #roc_auc = roc_auc_score(y_test, y_pred, multi_class='ovr')

Finalmente imprimimos nuestra ejecución del modelo

print('Curva ROC - AUC del modelo:')
#print(roc_auc) print('Precisión del modelo:', precision) print('Exactitud del modelo:', exactitud) print('Sensibilidad del modelo:', sensibilidad) print('Puntaje F1 del modelo:', puntajef1) #print('Curva ROC - AUC del modelo:', roc_auc)

Implementación mediante descenso del gradiente

El código base para implementación del descenso del gradiente, fue tomada de [4]

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Funciones para el cálculo del gradiente descendente, esta función retorna el valor w * x + b correspondiente al modelo lineal

```
def calcular_modelo(w,b,x):
    return w*x+b
```

Calculamos el error cuadrático medio entre el dato original (y) y el dato generado por el modelo (y_)

```
def calcular_error(y,y_):
    N = y.shape[0]
    error = np.sum((y-y_)**2)/N
    return error
```

Algoritmo del gradiente descendente para minimizar el error cuadrático medio.

```
def gradiente_descendente(w_, b_, alpha,
   x, y):
    N = x.shape[0]       # Cantidad de
       datos
```

La siguiente función calcula de los gradientes (derivadas) de la función de error con respecto a los parámetros "w" y "b"

```
dw = -(2/N)*np.sum(x*(y-(w_*x+b_)))
db = -(2/N)*np.sum(y-(w_*x+b_))
```

Actualizar los pesos usando la fórmula del gradiente descendente.

```
w = w_ - alpha*dw
b = b_ - alpha*db
return w, b
```

Modelo de entrenamiento

Una vez definido nuestro modelo y el método de aprendizaje´ configurado, se puede entrenar.

Para esto podemos entrenar o ajustar el modelo a los datos de entrenamiento disponibles invocando el método fit() del modelo:

model:f it(x train; y train; batch size = 100; epochs =5)

En los primeros dos argumentos, hemos indicado los datos con los que formaremos el modelo en forma de matrices Numpy. El tercer parámetro indicamos el número de datos que utilizaremos para entrenarlo y el ultimo parámetro indicamos las épocas de entrenamiento.

Este es el método que, consume mayores recursos en cuanto a tiempo. tflearn nos permite ver su progreso usando el argumento detallado (por defecto, igual a 1), además de indicar´ un estimado de cuánto dura cada época: ver figura 7, la cual´ presenta en color azul el valor de perdida y en a amarillo el valor de aciertos de la red.:

La figura 1. Indica el rango en el cual un estudiante reprueba o no una asignatura indicando que un valor mayor de 0.5 la materia es reprobada y un valor menor de 0.5 la materia es aprobada.

Figura 1. Grafica de Regresión logística tomado de https://aprendeia.com/algoritmo-regresion-logistica-machinelearning-teoria/ el día 13 de agosto de 2023

Resultados:

Obtendríamos lo siguiente:

Matriz de Confusión:

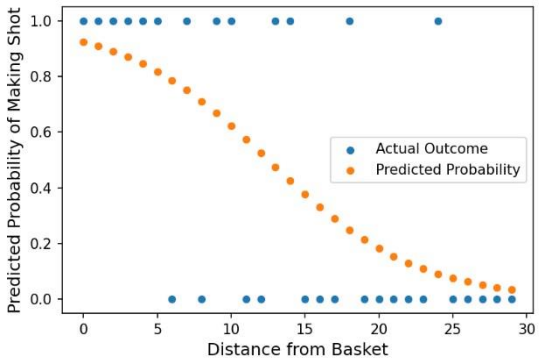


Figura 2. Regresión logística distancia vs aciertos

Figura 3. Matriz de Confusión datos de autor

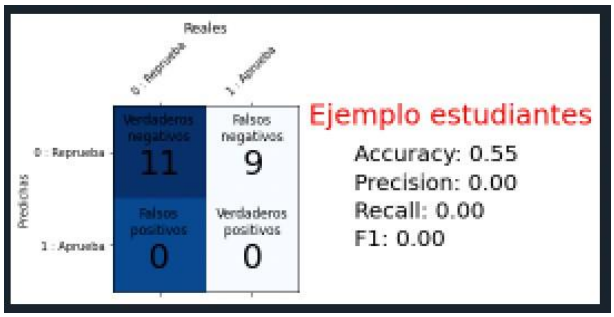


Figura 4. Métricas Accuracy, precisión, recall y F1 datos de autor

	0	1
0	0.402378	0.597622
1	0.402378	0.597622
2	0.409648	0.590352
3	0.409648	0.590352
4	0.409648	0.590352
5	0.40717	0.59283

Figura 5. Predicciones hechas con respecto al puntaje ICFES segundo grupo icfes bajo

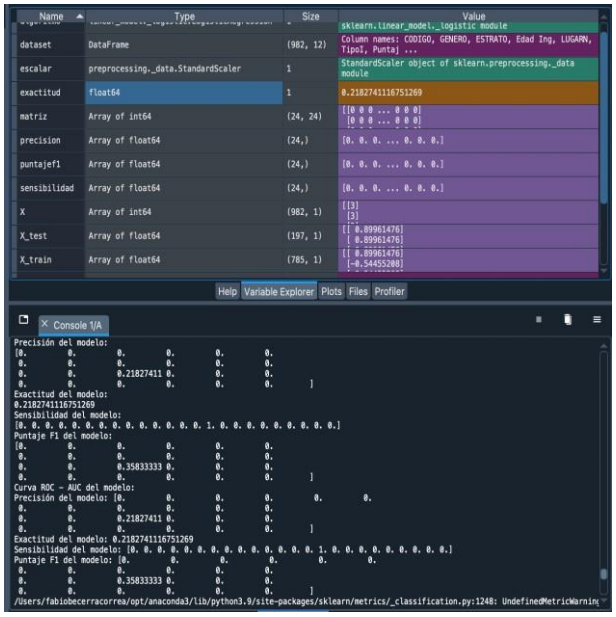
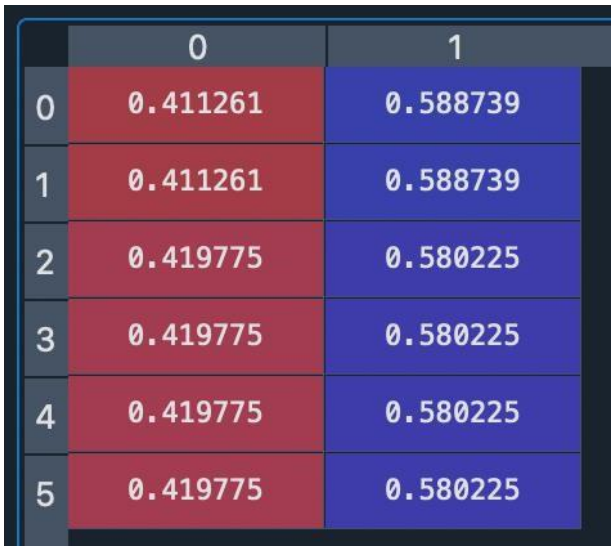


Figura 6. Resultados Experimentales datos de autor



	0	1
0	0.411261	0.588739
1	0.411261	0.588739
2	0.419775	0.580225
3	0.419775	0.580225
4	0.419775	0.580225
5	0.419775	0.580225

Figura 7. Predicciones hechas con respecto al puntaje ICFES segundo grupo icfes Alto

En las figuras, la primera columna corresponde a la probabilidad que el estudiante pierda la materia y la segunda que el estudiante apruebe la materia Modelo:

Como podemos observar un estudiante que tenga un ICFES de 471 puntos tiene 0.411261% de reprobar la materia y un 588739% de aprobarla con un icfes de 474.39 puntos.

El cual fue entrenado con 40000 épocas. Nuestro modelo tiene una pendiente de cero y cortara al eje Y en 17.3.

Trabajos futuros

Como trabajos futuros, se propone desarrollar, comparar los resultados obtenidos de implementar una red neuronal artificial multicapa y regresión logística. Igualmente utilizar regresión lineal, regresión polinomial, regresión escalonada, Regresión de crestas, Regresión de lazo y Regresión de ElasticNet. Igualmente podemos utilizar técnicas de minería de datos, que permitan optimizar la base datos de estudiantes para la limpieza de datos. Como ultimo proponemos explorar otras técnicas de machine learning como los Árboles de decisión.

Conclusiones

Nosotros hemos implementado el prototipo para predecir la probabilidad de que un estudiante de estudiante repruebe la materia de cálculo en primer semestre en la UDFJC, mediante el modelo de regresión logística, utilizando el método de mínimos cuadrados, implementado en Python utilizando la base de datos de la universidad distrital.

Evaluar los resultados es la parte más importante de esta investigación, ya que demuestra que tan preciso fue nuestro modelo y cuanto está cerca (lejos) de sus objetivos. En nuestro trabajo evaluaremos los modelos mediante: El error cuadrático medio, La matriz de confusión, La exactitud del modelo, Mean Squere error (MSE), La raíz del MSE, Root Mean Square Error (RMSE) y el coeficiente de determinación (R2).

Como podemos observar la precisión de nuestro modelo es 0.9641602272308584 lo cual indica que: Cuando predice sí, la frecuencia en que es correcto es de 0.97, nuestro modelo es muy preciso. Respecto a la exactitud del modelo con un valor de 0.9616633333333333, nos indica que cuando en realidad es un sí, la frecuencia predice un si es altísima.

La puntuación F1 es la media armónica de la precisión y exhaustividad, donde la puntuación de la F1 alcanza su mejor valor en 1 (precisión y exhaustividades perfectas) y el peor en 0. Cuando es no, ¿con que frecuencia predice el no? Tasa negativa real = TN/no real. Es la verdadera tasa negativa o la proporción de verdaderos negativos a todo lo que debería haber sido clasificado como negativo. La pérdida logarítmica aumenta a medida que la probabilidad predicha se aleja de la etiqueta real. El objetivo de cualquier modelo de aprendizaje automático es minimizar este valor. Por lo tanto, una perdida logarítmica menor es mejor, con un modelo perfecto teniendo una perdida logarítmica de 0. en nuestro modelo la perdida logarítmica es de 1.32, la cual es muy baja.

Referencias

[1] Regresión lineal y mínimos cuadrados (15 de Marzo de 2021). [Online Video]. Available url: https://www:youtube:com=watch?v = k964uNn3l0

[2] Entendiendo la regresión logística consultado el 15 agosto de 2023 https://www.datasource.ai/es/data-science-articles/entendiendo-la-regresion-logistica regresión Lineal con Scikit Learn (14 de Marzo de 2021). [Online Video]. Available url = https: //aprendeia:com/regression-lineal-con-scikit-learn/

[7] John Burkardt. Regression Linear Datasets (14 de Marzo de 2021). Available url = https://people:sc:fsu:edu/ jburkardt/datasets/regression/regression.html

[8] Anurag Gupta. Using Python to visualize COVID-19 projections (14 de Marzo de 2021). Available url = https: //opensource:com/article/20/4/python-data-covid-19

[9] Prototipo de una red neuronal artificial para la detección de falencias académicas en primer semestre Facultad Tecnológica (14 de Marzo de 2021). Available url = https://www.udistrital.edu.co/inicio

[10] Minsky, Marvin and Papert, Seymour ,"An introduction to computational geometry", Cambridge tiass., HIT, year=1969

[11] Rosenblatt, Frank, "he perceptron: a probabilistic model for information storage and organization in the brain." Psychological review, volume=65, number=6, pages=386, year=1958.

[12] Rumelhart, David E and Hinton, Geoffrey E and Williams, Ronald J, Learning representations by back-propagating errors journal=nature, volume=323, number=6088, pages=533–536, year=1986, publisher=Nature Publishing Group

[13] Wikipedia.org, Python, year = 2020, url = URL https://es.wikipedia.org/wiki/Python, note/"[Web; accedido el 26-04-2020]"

[14] Jordi Torres, "Red neuronal en Keras, guia practica,",year = septiembre 19, 2018,url = URL http://sitiobigdata.com/2018/09/19/red-neuronal-enkeras/, note = [Web; accedido el 26-04-2020]

[15] "Keras.io," year = 2020, url = URL https://keras.io/,note/ [Web accedido el 26-04-2020]

[16] author = Wikipedia.org ,title = Tflearn, year = 2020, url = URL https://es.wikipedia.org/wiki/Tflearn,note = "[Web; accedido el 26-04-2020]"

[17] author = Wikipedia.org ,title = Keras, year = 2020, url = URL https://es.wikipedia.org/wiki/Keras, note = "[Web; accedido el 26-04-2020]"

[18] author = Guillermo Westreicher,title = economipedia, year = 2021, url = URL https://economipedia.com/definiciones/modelo-logit.html [Web; accedido el 26-04-2023]"

[19] Congreso internacional de computación CICOM201. consultado el 15 agosto de 2023 https://comunidad.udistrital.edu.co/cicom2021/organizacion/