Universidad del Norte

Departamento de Ingeniería de Sistemas y Computación

Estructura de Datos I

Laboratorio 3

Prof. Sebastián Racedo

Integrantes

Juan Esteban Albis

Mariana Castañeda

Shadia Jaafar

Kevin Jaimes

Resumen

En este documento se encuentran los puntos más relevantes del Laboratorio. En primer lugar, una breve introducción, luego, el Diagrama de Clases (UML) el cual registra de forma gráfica la estructura del Proyecto según las clases, sus respectivos atributos, métodos, y las relaciones entre los objetos. En el siguiente apartado, se explica la Limpieza de Los Datos, la selección de las variables de interés y el tratamiento de los nulos. Luego, se expone los métodos elaborados, junto a sus funciones particulares y el razonamiento usado para la implementación de ciertas funciones en estos mismos.

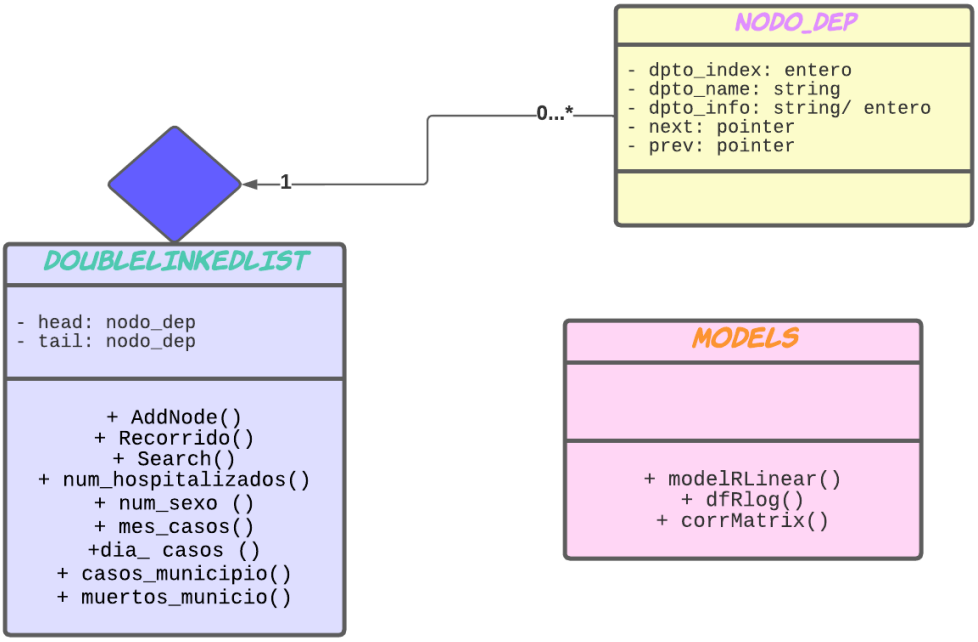
Tabla de Contenido

1. Introducción

El Presente laboratorio tiene como objetivo afianzar los conocimientos en Las Estructuras enseñadas en clase junto al Manejo de Datos, de manera que es de suma importancia el entendimiento y la oportuna limpieza de datos para trabajar con información de utilidad.

En esta ocasión, se empleó el a archivo “DENGUE 2010-2020.xlsx”, el cual presenta los casos de Dengue que se registraron en el periodo de 2010 a 2020, junto a más información relevante relacionada a los registros de la Enfermedad mencionada, extraída de la Ficha de Notificación Individual, la cual se tiene en cuenta para la comprensión de las variables. Por otro lado, para la codificación de cada punto, se implementará el Lenguaje de Programación, Python, a través de un Jupyter Notebook, ya que este presenta La Librería pandas, herramienta de análisis y manipulación fácil de datos.

2. Diagrama de Clases UML



3. Limpieza del archivo

Para la limpieza del archivo “DENGUE 2010-2020.xlsx”, se tuvieron en cuenta los siguientes pasos:

3.1 Variables relevantes

En primer lugar, se seleccionaron las variables de interés. Estas fueron:

- PAC_HOS: paciente hospitalizado (tipo entero)
  1.0 = Si
  2.0 = No
- SEXO: sexo del paciente (tipo string)

  F = Femenino

  M = Masculino

- Municipio_ocurrencia: municipio donde ocurrió el contagio (tipo string)
- CON_FIN: condición final del paciente (tipo entero)
  1.0 = Vivo
  2.0 = Muerto
- FEC_NOT: fecha de notificación (tipo string)
- ANO: año (tipo entero)
- EDAD: edad del paciente (tipo entero)
- Departamento_ocurrencia: departamento donde ocurrió el contagio (tipo string)

Nota: La interpretación de cada variable se realizó por medio de la Ficha Notificación Individual

3.2 Tratamiento de los datos nulos:

Por otro lado, uno de los principales inconvenientes fue la presencia de datos nulos. Por lo que primero se creó el archivo “df_clean.csv” y se calculó la proporción de los datos nulos con respecto al total, con el fin de comprobar si era representativo o no. En este caso, la cantidad de nulos no lo era, por lo que se decidió eliminar las filas que los contuvieran.

4. Métodos principales

Entre Departamentos

Para los métodos que arrojan información de un Departamento especifico, como primer paso deberá recorrer la lista doblemente enlazada de Departamentos y encontrar el nodo que corresponde al Departamento de interés con ayuda de un método *search ()* (este sigue la lógica de otros métodos de búsqueda realizados en clase), seguidamente tendrá que seleccionar del DataFrame (este contiene toda la información del departamento) que tiene como atributo, la columna(s) necesarias según sea el método.

Teniendo esto en cuenta, se procederá a explicar lo particular en cada método.

- *num_hospitalizados ()* * Cantidad de casos hospitalizados por Dpto.

  Del DataFrame del Departamento de interés, se "extrae" un nuevo DataFrame que contenga los datos de la columna "PAC_HOS" (columna de pacientes hospitalizados) iguales a 1 (según La Ficha 1 corresponde a los Pacientes Hospitalizados). Seguidamente con la función len () se calcula el número de filas que contiene el DataFrame que corresponderá al número de hospitalizados.

- *num_sexo ()* *Cantidad de Hombres/ Mujeres registrados

  En este caso se "extrae" del DataFrame del Departamento, dos DatraFrames nuevos, uno que almacenará "M" y el otro "F" de la columna "SEXO" indicando masculino y femenino respectivamente. Con len () obtendremos el tamaño de cada uno, indicando el número de mujeres y hombres por Departamento.

Para los siguientes métodos es necesario importa La Librería datetime y calendar (herramientas para la manipulación de fechas)

- *mes_casos ()* *Mes con mayor número de casos registrados por departamento

Se agrupa el DataFrame del Departamento según la columna "FEC_HOS" con ayuda de la función .groupby(), además se busca la frecuencia de cada agrupación con .value_counts que por defecto nos entregara la frecuencia descendentemente.

pd.to_datetime (de la columna "FEC_HOS").dt.month, retornara un objerto tipo fecha (en este caso, mes) el cual será una nueva columna llamada MES.

MES se convierte a lista (values.tolist) para guardar la cada mes y su frecuencia, pero solo será de interés la primera posición (la de mayor frecuencia). La primera posición de la lista que

corresponde al mes con mayor frecuencia, se le aplicara el método calendar.month_name[lista] el cual retornará según el nombre del mes según el número.

- *dia_ casos ()* *Día con mayor número de casos registrados por departamento

Igual que *mes_casos ()*, se agrupa el DataFrame del Departamento según la columna "FEC_HOS" con.groupby(), y se busca la frecuencia de cada agrupación con .value_

En este cado usaremos pd.to_datetime (de la columna "FEC_HOS").dt.day, que retornara el dia en objeto de tipo fecha y será una nueva columna "DIA", esta se convierte a lista para guardar la cada dia y su frecuencia, pero solo será de interés la primera posición. En el Print retornara la primera posición de la lista (el dia con mayor frecuencia).

Entre Municipios

Para los métodos que arrojan información sobre los Municipios, se sigue el razonamiento inicial de los métodos para Departamentos, de manera que se encuentra el nodo del departamento que contenga en el municipio de intereses y se posicione en la columna del DataFrame según el departamento que corresponda al municipio y luego específicamente en la columna del municipio.

Teniendo esto en cuenta, se procederá a explicar lo particular en cada método.

- *casos_municipio ()* * Municipio con mayor número de casos

Teniendo en cuenta que se trata una Enfermedad de Notificación Obligatoria, se puede asegurar que el número de columnas registradas en el DataFrame es igual al numero de casos.

Del DataFrame del departamento de interés, se hace una lista de los municipios y con la función .valeu_counts se obtiene el numero de casos registrados.

- *muertos_municipio ()* * Municipio con mayor número de muertos

Del DataFrame del Departamento, se agrupa los datos con .groupby por municipio según la columna de "CON_FIN" y con .value_counts() guardamos la frecuencia, luego se elabora una lista de los datos en la columna "CON_FIN" iguales a 2, que representan los muertos según La Ficha. Esta se ordeno ascendentemente para obtener el Municipio con más muertos en la primera posición de la lista.

5. **Modelo de predicción**

Regresión lineal vs. Regresión logística

La regresión lineal le brinda una salida continua, pero la regresión logística proporciona una salida constante. Un ejemplo de la salida continua es el precio de la vivienda y el precio de las acciones. El ejemplo de la salida discreta es predecir si un paciente tiene cáncer o no, predecir si el cliente abandonará. La regresión lineal se estima utilizando mínimos cuadrados ordinarios (OLS), mientras que la regresión logística se estima utilizando el enfoque de estimación de máxima verosimilitud (MLE).

Regresión Lineal

Un modelo lineal genera predicciones al hacer una suma ponderada de sus variables, más una constante.

En la ecuación:

- y es la variable para predecir.
- n es el número de variables predictoras.
- Las x representan el valor de las variables predictoras.
- Las b son los parámetros del modelo, los cuales son los que intentamos ajustar para reducir el error o la distancia entre los valores.

Coeficiente de regresión

En regresión lineal simple, si el coeficiente de x es positivo, entonces podemos concluir que la relación entre las variables independiente y dependiente es positiva.

Ahora, si el coeficiente de x es negativo, entonces podemos decir que la relación entre las variables independiente y dependiente es negativa.

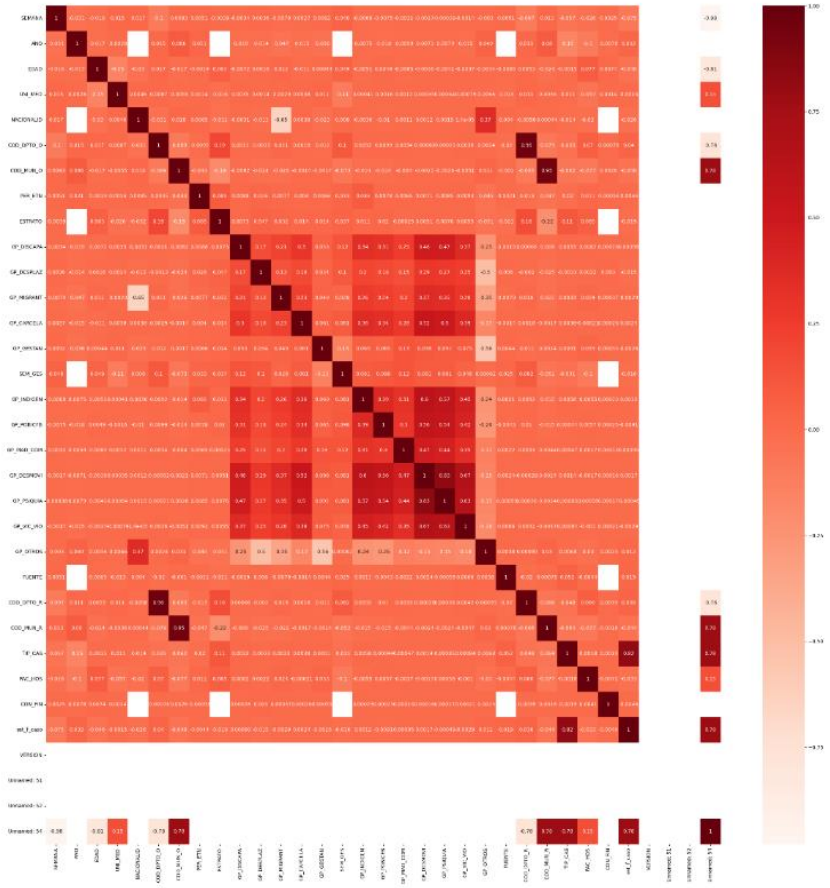
Antes de comenzar con la práctica de la regresión lineal y regresión logística en Python, exploremos el conjunto de datos. Usaremos el conjunto de datos sobre casos de dengue, con 797.167 filas y 53 columnas.

FEC_NOT	SEMANA	ANO	EDAD	UNI_MED	NACIONALID	NOM_NACIONALIDAD	SEXO	COD_DPTO_O	COD_MUN_O	VEREDA
##########	8	2020	26	1	170	COLOMBIA	M	76	892	SIN DATO
##########	23	2020	7	1	170	COLOMBIA	M	76	364	SIN DATO
##########	28	2020	19	1	170	COLOMBIA	M	25	488	SIN DATO
4/01/2020	1	2020	43	1	170	COLOMBIA	M	73	283	SIN DATO
##########	11	2020	39	1	170	COLOMBIA	M	68	533	PALMARITO
##########	24	2020	3	1	170	COLOMBIA	F	23	189	SIN DATO
##########	12	2020	15	1	170	COLOMBIA	F	68	533	SIN DATO
6/02/2020	5	2020	2	1	170	COLOMBIA	M	70	1	SIN DATO
6/02/2020	5	2020	9	1	170	COLOMBIA	M	70	1	SIN DATO
6/10/2020	41	2020	43	1	170	COLOMBIA	F	25	875	SIN DATO
##########	2	2020	24	1	170	COLOMBIA	F	70	1	SIN DATO
##########	8	2020	3	2	170	COLOMBIA	F	70	1	SIN DATO
##########	3	2020	25	1	170	COLOMBIA	M	76	834	SIN DATO
7/10/2020	40	2020	1	1	170	COLOMBIA	M	76	834	SIN DATO
##########	2	2020	6	1	170	COLOMBIA	M	73	678	MAL NOMBRE
##########	16	2020	76	1	170	COLOMBIA	M	76	834	SIN DATO
##########	7	2020	66	1	170	COLOMBIA	M	76	834	SIN DATO
##########	17	2020	19	1	170	COLOMBIA	M	25	488	SIN DATO
##########	22	2020	78	1	170	COLOMBIA	F	41	503	SIN DATO
##########	9	2020	19	1	170	COLOMBIA	M	76	828	SIN DATO
9/05/2020	19	2020	42	1	170	COLOMBIA	F	76	828	SIN DATO

Model Building

Simple Linear Regression

Lo primero es identificar las variables que se les aplicara el modelo. Para esto hicimos una matriz de correlación.



Después de analizar la matriz, vimos que las variables que tienen mayor correlación son entre ellas mismas y las variables dicotómicas, las cuales para una regresión lineal no son suficientes para llegar a una buena conclusión estadística. De acuerdo con lo anterior, llegamos a la conclusión de que una regresión lineal para estos datos será ineficiente y no se tendrá el resultado deseado. Sin embargo, hicimos pruebas para verificar nuestra hipótesis.

In [360…

```
class models:

    def modelRLinear(self):
        x = data_1[["EDAD"]]
        y = data_1["SEMANA"]
        # Split the data into training/testing sets
        x_train = x[:-20]
        x_test = x[-20:]

        # Split the targets into training/testing sets
        y_train = y[:-20]
        y_test = y[-20:]

        regr = linear_model.LinearRegression()

        # Train the model using the training sets
        regr.fit(x_train, y_train)

        # Make predictions using the testing set
        y_pred = regr.predict(x_test)

        # The coefficients
        print("Coefficients: \n", regr.coef_)
        # The mean squared error
        print("Mean squared error: %.2f" % mean_squared_error(y_test, y_pred))
        # The coefficient of determination: 1 is perfect prediction
        print("Coefficient of determination: %.2f" % r2_score(y_test, y_pred))

        # Plot outputs
        plt.scatter(x_test, y_test, color="black")
        plt.plot(x_test, y_pred, color="blue", linewidth=3)

        plt.xticks(())
        plt.yticks(())

        plt.show()
```

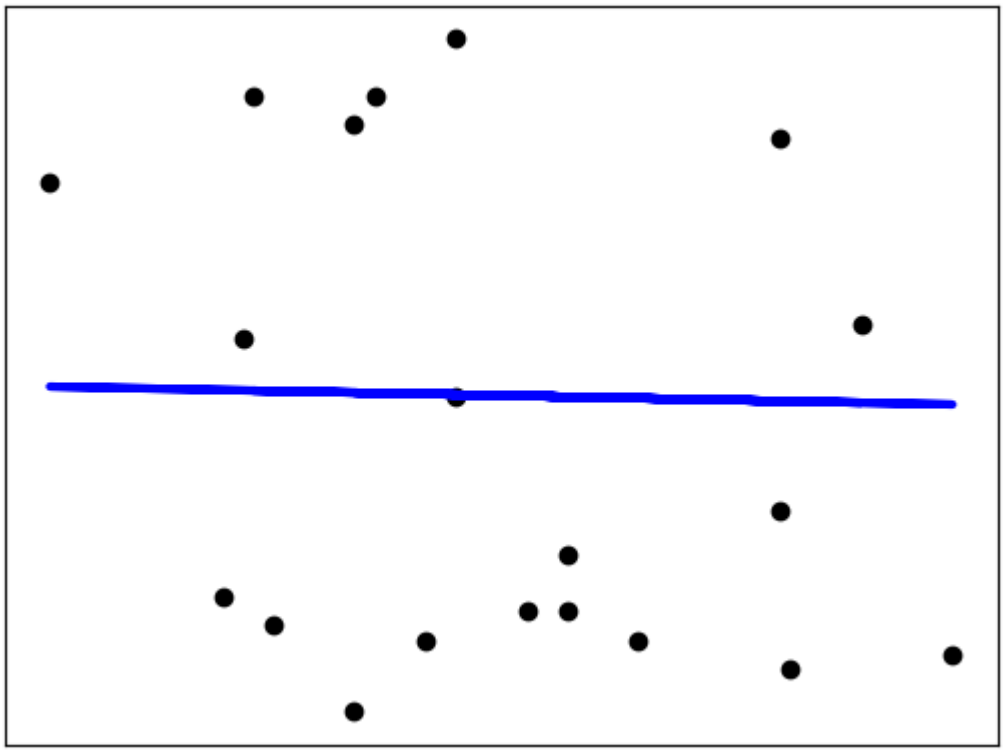
Primero elegimos las dos variables que necesitamos para el modelo y las predicciones. (Hicimos pruebas con todas las variables numéricas que no sean dicotómicas.)

En este caso estaremos utilizando “EDAD” y “SEMANA”. Después de esto separamos ambas variables en training/test sets, lo que hace el programa es entrenar y poner a prueba un número de datos para poder hacer la predicción con más precisión.

Haremos las predicciones utilizando los sets de entrenamiento, calcularemos el coeficiente de correlación, la media al cuadrado y el coeficiente de determinación.

```
Coefficients:
 [-0.01410052]
Mean squared error: 262.15
Coefficient of determination: -0.01
```

Observando los resultados, podemos decir con certeza que no tienen correlación entre ellas, ya que los valores del coeficiente y el coeficiente de determinación se aproximan a 0, esto significa que las variables estudiadas no tienen relación lineal entre ellas. Para poder verlo más detallado:



Viendo la dispersión de los datos, es claro que no tienen ningún tipo de relación. Por lo tanto, no servirá para nuestro estudio.

Regresión Logística

Las técnicas de clasificación son una parte esencial de las aplicaciones de aprendizaje automático y minería de datos. Aproximadamente el 70% de los problemas en Data Science son problemas de clasificación. Hay muchos problemas de clasificación disponibles, pero la regresión logística es común y es un método de regresión útil para resolver el problema de clasificación binaria. Otra categoría de clasificación es la clasificación multinomial, que maneja los problemas en los que hay varias clases presentes en la variable de destino.

La regresión logística se usa normalmente para estimar la probabilidad de que un evento o instancia pertenezca a cierta clase (clasificación).

Si la probabilidad estimada es mayor a 50% entonces el modelo la clasifica como que, si pertenece a esa clase, por ejemplo, queremos saber si un objeto es rojo y el modelo lo capta como 70% rojo, entonces lo clasifica como rojo.

Por esto mismo funciona para saber si cierta instancia es de una clase o no es de esa clase, por lo que es un clasificador binario.

¿Como funciona la regresión logística?

Como una regresión lineal, una regresión logística no regresa un modelo para hacer predicciones con diferentes valores, si no que te regresa un modelo donde intenta clasificar si un objeto pertenece a cierta instancia o no.

Logistic Regression

Como hicimos con el modelo anterior, elegimos las variables que necesitaremos. En este caso para la variable x, utilizaremos más de una variable para hacer un modelo preciso.

Para este modelo hicimos un análisis de las variables que ayudaban con la relación entre ellas y la de estudio (en este caso, grupo indígenas). Se debe dividir las columnas dadas en dos tipos de variables dependientes (o variables de destino) y variables independientes (o variables de características).

```
def modeloRLogistic(self):

    vari = ["UNI_MED","EDAD","COD_DPTO_O", "COD_MUN_O","PER_ETN"] # Variables use for x in the Logistic Regression
    x = numericos[vari]
    y = numericos.GP_INDIGEN
    clf = LogisticRegression()

    # split X and y into training and testing sets
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)

    clf.fit(x_train,y_train)
    score = clf.score(x_train, y_train)
    y_pred = clf.predict(x_test)

    y_pred_proba = clf.predict_proba(x_test)[::,1]
    fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba,pos_label= 2)
    auc = metrics.roc_auc_score(y_test, y_pred_proba)
    plt.plot(fpr,tpr,label="Data, auc="+str(auc))
    plt.legend(loc=4)
    plt.show()
```

División de datos:

Para comprender el rendimiento del modelo, dividir el conjunto de datos en un conjunto de entrenamiento y un conjunto de prueba.

Dividamos el conjunto de datos usando la función train_test_split(). Debe pasar 3 características de parámetros, objetivo y tamaño de test_set.

El conjunto de datos se divide en dos partes en una proporción de 75:25. Significa que el 75 % de los datos se usarán para el entrenamiento del modelo y el 25 % para la prueba del modelo.

Desarrollo y predicción de modelos

Primero, importe el módulo Logistic Regression y cree un objeto clasificador Logistic Regression utilizando la función LogisticRegression().

Luego, ajuste su modelo en el conjunto de entrenamiento usando fit() y realice la predicción en el conjunto de prueba usando predict().

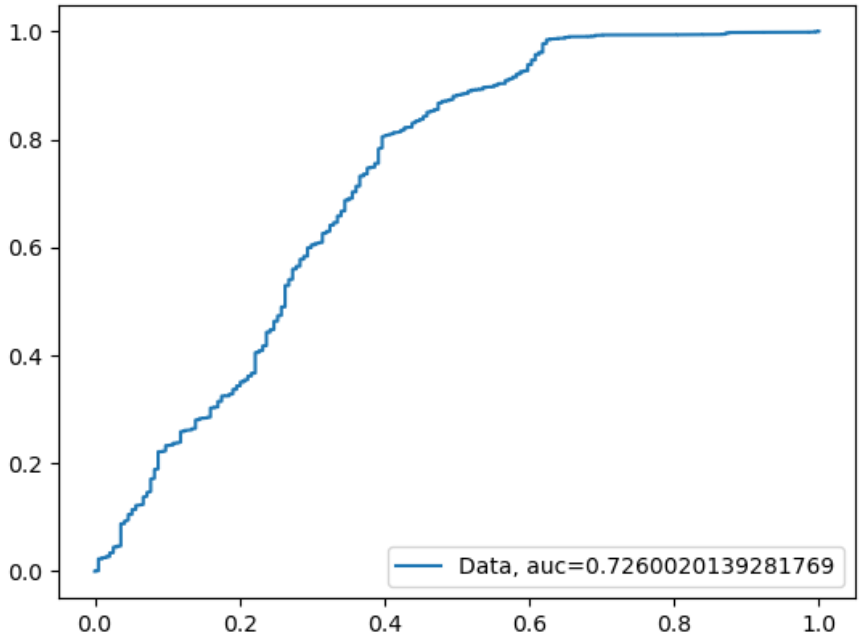
	GP_INDIGEN	predict	Ind	No Ind
0	2.0	2.0	0.000494	0.999506
1	2.0	2.0	0.001552	0.998448
2	2.0	2.0	0.000524	0.999476
3	2.0	2.0	0.000525	0.999475
4	2.0	2.0	0.000599	0.999401
...	...	...	...	...
797161	2.0	2.0	0.000002	0.999998
797162	2.0	2.0	0.000002	0.999998
797163	2.0	2.0	0.001563	0.998437
797164	2.0	2.0	0.000904	0.999096
797165	2.0	2.0	0.000900	0.999100

[797166 rows x 4 columns]

Aquí podemos observar la variable de estudio y las predicciones de esta. Se hizo con la probabilidad de cada uno, para poder saber si la predicción fue acertada o si hubo algún tipo de problema al clasificar si es o no del grupo indígena.

Precisión: la precisión se trata de ser preciso, es decir, qué tan preciso es el modelo. En otras palabras, podemos decir, cuando un modelo hace una predicción, con qué frecuencia es correcta. En nuestro caso de predicción, cuando el modelo de regresión logística predijo que el individuo es del grupo indígena, tal individuo no pertenece a este grupo el 99 % del tiempo.

Curva ROC La curva de características operativas del receptor (ROC) es un gráfico de la tasa de verdaderos positivos frente a la tasa de falsos positivos. Muestra el compromiso entre sensibilidad y especificidad.



La puntuación AUC para el caso es 0,72. La puntuación AUC 1 representa un clasificador perfecto y 0,5 representa un clasificador incompetente.