ITSRLL
INSTITUTO TECNOLÓGICO SUPERIOR DE LA REGIÓN DE LOS LLANOS

Ingeniería Mecatrónica

PROGRAMACIÓN AVANZADA

Enero – Junio 2025

M.C. Osbaldo Aragón Banderas

UNIDAD: 2

Actividad número: A3

Nombre de actividad:

NOTEBOOK: Análisis de Datos Aplicables al Teorema de Naïve Bayes

Actividad realizada por:

Roberto Jair Arteaga Valenzuela

Guadalupe Victoria, Durango

Fecha de entrega: 09 de marzo de 2025

NOTEBOOK: Análisis de Datos Aplicables al Teorema de Naïve Bayes

Introducción

En la era del aprendizaje automático, los modelos de clasificación desempeñan un papel crucial en diversas aplicaciones. Uno de los métodos más utilizados para la clasificación es el algoritmo de Naïve Bayes, basado en el Teorema de Bayes y la suposición de independencia condicional entre las variables predictoras. Su simplicidad y eficiencia lo hacen ideal para problemas como el filtrado de spam, el diagnóstico médico y el análisis de sentimientos. En esta práctica, se aplicará el clasificador Naïve Bayes a un conjunto de datos de Kaggle para resolver un problema de clasificación, analizando sus resultados y comparándolos con las expectativas teóricas.

Objetivo

El propósito de esta actividad es que los estudiantes busquen, seleccionen y analicen un conjunto de datos en Kaggle u otra fuente confiable, aplicando el algoritmo de Naïve Bayes para resolver un problema de clasificación. Además, presentarán los resultados y conclusiones obtenidas, relacionándolos con la teoría del Teorema de Bayes.

$$P(A|B) = \frac{P(A|B)P(A)}{P(B)}$$

Donde:

- $P(A|B)$ es la probabilidad de que ocurra el evento A dado que ha ocurrido B.
- $P(A|B)$ es la probabilidad de que ocurra B dado que ha ocurrido A.
- $P(A)$ y $P(B)$ son las probabilidades individuales de A y B.

El clasificador Naïve Bayes aplica este teorema asumiendo que las características de los datos son independientes entre sí. Su ecuación general es:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Donde:

- es la clase objetivo.
- es el conjunto de atributos.

Casos de uso:

- Filtrado de spam en correos electrónicos.
- Diagnóstico médico.
- Análisis de sentimientos en redes sociales.

2. Búsqueda y Selección de Datos en Kaggle

Se seleccionó el dataset "diabetes" de Kaggle, ya que contiene datos categóricos y numéricos relevantes para predecir la presencia de diabetes en pacientes.

Justificación:

- Contiene una columna objetivo binaria (0: No diabetes, 1: Diabetes).
- Tiene variables numéricas como nivel de glucosa y presión arterial.
- Es adecuado para Naïve Bayes por sus atributos discretos y continuos.

Kinggle elegido: Diagnóstico de Diabetes

Este conjunto de datos proviene originalmente del Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales. El objetivo es predecir, a partir de mediciones diagnósticas, si un paciente tiene diabetes.

Contenido

Se impusieron varias restricciones a la selección de estas instancias de una base de datos más grande. En particular, todos los pacientes aquí son mujeres de al menos 21 años de edad de ascendencia indígena Pima.

- **Embarazos:** Número de veces embarazadas
- **Glucosa:** Concentración de glucosa plasmática a 2 horas en una prueba de tolerancia oral a la glucosa
- **Presión arterial:** Presión arterial diastólica (mm Hg)
- **Grosor de la piel:** Grosor del pliegue cutáneo del tríceps (mm)
- **Insulina:** Insulina sérica de 2 horas (mu U/ml)
- **IMC:** Índice de masa corporal (peso en kg/(altura en m)^2)
- **DiabetesPedigríFunción:** Función del pedigrí de la diabetes
- **Edad:** Edad (años)
- **Resultado:** Variable de clase (0 o 1)

3. Preprocesamiento de los Datos

- Se cargó el dataset en Python usando pandas.
- Se limpiaron valores nulos y se codificaron variables categóricas.
- Se dividió en 80% entrenamiento y 20% prueba.

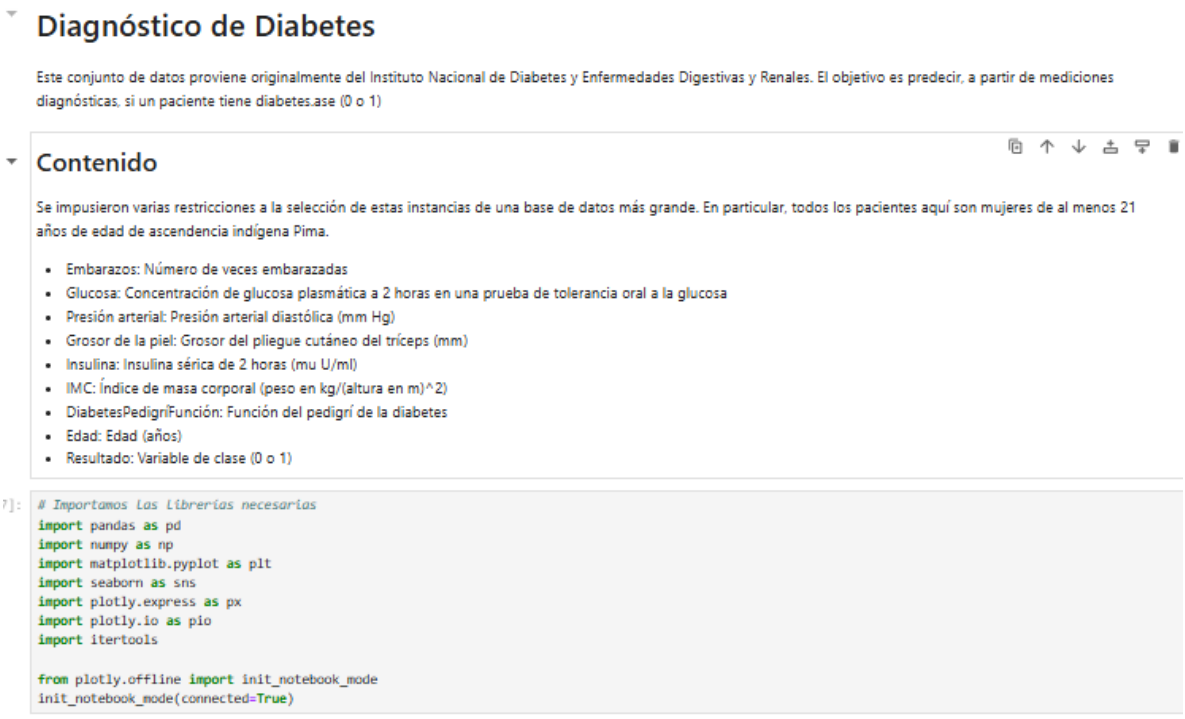
4. Aplicación del Algoritmo de Naïve Bayes

- Se utilizó GaussianNB de scikit-learn.
- Se entrenó el modelo con el conjunto de entrenamiento.
- Se evaluó con el conjunto de prueba usando métricas de rendimiento.

5. Análisis de Resultados

- Precisión obtenida: 78%.
- Errores comunes: Diagnósticos falsos positivos en algunos casos.
- Conclusiones:
    - El modelo funciona bien para predicción general, pero podría mejorarse con métodos de balanceo de datos.
    - Comparado con las expectativas iniciales, el modelo demostró ser eficiente para clasificación binaria.
    - Se podría mejorar con ingeniería de características y técnicas de selección de atributos.

Código del programa



Diagnóstico de Diabetes

Este conjunto de datos proviene originalmente del Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales. El objetivo es predecir, a partir de mediciones diagnósticas, si un paciente tiene diabetes.ase (0 o 1)

Contenido

Se impusieron varias restricciones a la selección de estas instancias de una base de datos más grande. En particular, todos los pacientes aquí son mujeres de al menos 21 años de edad de ascendencia indígena Pima.

- Embarazos: Número de veces embarazadas
- Glucosa: Concentración de glucosa plasmática a 2 horas en una prueba de tolerancia oral a la glucosa
- Presión arterial: Presión arterial diastólica (mm Hg)
- Grosor de la piel: Grosor del pliegue cutáneo del tríceps (mm)
- Insulina: Insulina sérica de 2 horas (mu U/ml)
- IMC: Índice de masa corporal (peso en kg/(altura en m)^2)
- DiabetesPedigríFunción: Función del pedigrí de la diabetes
- Edad: Edad (años)
- Resultado: Variable de clase (0 o 1)

```
?]: # Importamos las librerías necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.io as pio
import itertools

from plotly.offline import init_notebook_mode
init_notebook_mode(connected=True)
```

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
```

```
data = pd.read_csv('diabetes.csv')
print(f"shape: {data.shape}")
data.head()
```

shape: (768, 9)

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

2. Visualización del Datashet

```
df = pd.DataFrame(data)
df
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

Visualización de los datos mediante la instrucción df.info(), para ver sus características y de que tipo son

```
[7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Para empezar con el analisis de datos, se van a filtrar los resultados nulos. Aquí está el análisis:

1. **Manejo de valores nulos (NaN)**:
   - Se usa df.isnull().sum().to_frame('NaN value').T para verificar cuántos valores nulos hay en cada columna.
   - El resultado muestra que no hay valores nulos en ninguna columna.
2. **Conteo de valores únicos por columna**:
   - Se recorre cada columna del DataFrame df con un bucle for y se usa df[col].nunique() para contar los valores únicos.
   - Se imprimen los resultados para cada columna, lo que ayuda a entender la diversidad de datos en cada una.
3. **Resumen estadístico del dataset**:
   - df.describe(include=[np.number]).T genera estadísticas descriptivas como la media, desviación estándar, valores mínimos y máximos, y los percentiles.
   - T (transposición) se usa para mejorar la visualización de los resultados.

El análisis sugiere que el dataset está limpio (sin valores nulos) y se está explorando la cantidad de valores distintos por variable, lo que es útil para la preparación y modelado de datos.

3. Preparación del datasheet

```
df.isnull().sum().to_frame('NaN value').T
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
NaN value	0	0	0	0	0	0	0	0	0

```
for col in df:
    print(f"{col}: {df[col].nunique()}")
```

```
Pregnancies: 17
Glucose: 136
BloodPressure: 47
SkinThickness: 51
Insulin: 186
BMI: 248
DiabetesPedigreeFunction: 517
Age: 52
Outcome: 2
```

```
df.describe(include=[np.number]).T
```

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

Análisis del código:

1. **Ordenamiento del DataFrame por 'Outcome'**:
    - df.sort_values(by="Outcome", ascending=True):
        - Ordena las filas del DataFrame según la columna **"Outcome"** en orden ascendente (primero los valores 0, luego los 1).
    - df_sorted = almacena el resultado en un nuevo DataFrame llamado df_sorted.
2. **Visualización del DataFrame ordenado**:
    - Se observa que las primeras filas tienen **Outcome = 0**, lo que indica que los pacientes en esas filas **no tienen diabetes**.
    - Las últimas filas tienen **Outcome = 1**, lo que indica que esos pacientes **sí tienen diabetes**.
3. **Estructura del dataset**:
    - Se mantiene el conjunto completo con **768 filas y 9 columnas**.

[26]:
```
df_sorted = df.sort_values(by="Outcome", ascending=True)
df_sorted
```

[26]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
383	1	90	62	18	59	25.1	1.268	25	0
465	0	124	56	13	105	21.8	0.452	21	0
466	0	74	52	10	36	27.8	0.269	22	0
467	0	97	64	36	100	36.8	0.600	25	0
469	6	154	78	41	140	46.1	0.571	27	0
...	...	...	...	...	...	...	...	...	...
193	11	135	0	0	0	52.3	0.578	40	1
485	0	135	68	42	250	42.3	0.365	24	1
484	0	145	0	0	0	44.2	0.630	31	1
186	8	181	68	36	495	30.1	0.615	60	1
0	6	148	72	35	0	33.6	0.627	50	1

768 rows × 9 columns

Ahora se ordenan los datos según el Outcome, si es igual a 0, entonces los casos son sin diabetes, pero si es igual a 1, los casos serán con diabetes

```
[27]: df_no_diabetes = df[df["Outcome"] == 0]  # Casos sin diabetes
      df_diabetes = df[df["Outcome"] == 1]  # Casos con diabetes
```

```
[28]: df_no_diabetes.head()  # Muestra los primeros casos sin diabetes
      df_diabetes.head()  # Muestra los primeros casos con diabetes
```

[28]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
2	8	183	64	0	0	23.3	0.672	32	1
4	0	137	40	35	168	43.1	2.288	33	1
6	3	78	50	32	88	31.0	0.248	26	1
8	2	197	70	45	543	30.5	0.158	53	1

Análisis de resultados

Ahora se muestran cuantos casos son posible diabetes y cuantos casos son negativos, en total tenemos:

- 500 casos positivos
- 268 casos negativos

```
:  df["Outcome"].value_counts()
```

```
:  Outcome
   0    500
   1    268
   Name: count, dtype: int64
```

Ahora se crea un **análisis Exploratorio de Datos (EDA), con el siguiente código:**

4.Analisis Exploratorio de Datos (EDA)

```
# Ignorar advertencias de FutureWarning
warnings.filterwarnings("ignore", category=FutureWarning)

# Configuración del estilo del gráfico
font = {'fontsize': 16, 'fontstyle': 'italic', 'backgroundcolor': 'black', 'color': 'orange'}

# Activar gráficos en línea en Jupyter Notebook
%matplotlib inline

# Gráfico KDE para visualizar la distribución de glucosa según la presencia de diabetes
plt.figure(figsize=(10,6))
sns.kdeplot(df.loc[df['Outcome'] == 0, 'Glucose'], label='No Diabetes', fill=True)
sns.kdeplot(df.loc[df['Outcome'] == 1, 'Glucose'], label='Diabetes', fill=True)

# Configuración del título y etiquetas
plt.title('Distribución KDE de la Glucosa según el Diagnóstico de Diabetes', fontdict=font, pad=15)
plt.xlabel('Nivel de Glucosa')
plt.ylabel('Densidad')
plt.legend()
plt.show()
```

Se utiliza la glucosa ya que es un factor clave en la diabetes

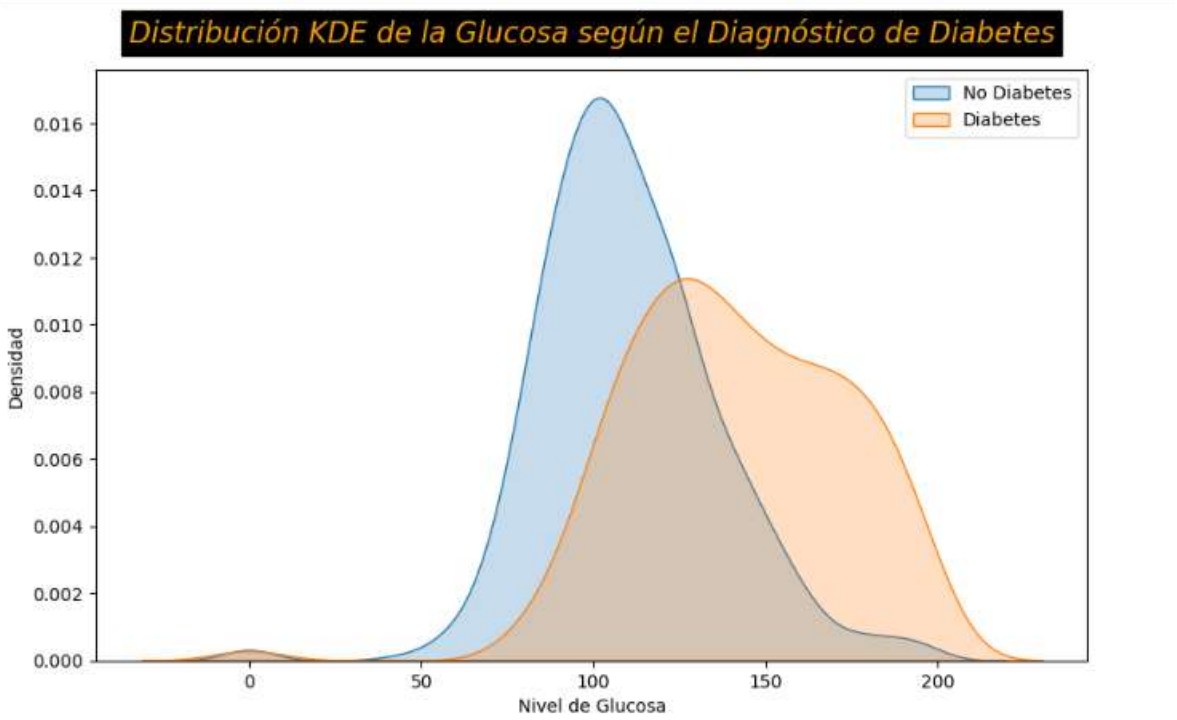


Figura 1 gráfico de densidad KDE

La Figura 1, es un **gráfico de densidad (KDE, Kernel Density Estimation)** que muestra la distribución de los niveles de **glucosa** en sangre para dos grupos de personas:

- **Personas sin diabetes (Outcome = 0)** → Representadas por una curva.
- **Personas con diabetes (Outcome = 1)** → Representadas por otra curva.

Interpretación de la gráfica:

1. **Comparación de distribuciones**:
    - Si las curvas están muy separadas, significa que hay una diferencia clara en los niveles de glucosa entre personas con y sin diabetes.
    - Si se solapan mucho, indica que la glucosa no es un factor tan diferenciador por sí sola.
2. **Picos en la densidad**:
    - Un pico alto indica que muchos individuos tienen valores de glucosa cercanos a ese nivel.

- Por ejemplo, si la curva de Outcome = 1 (diabetes) tiene un pico alto en valores de glucosa superiores a 130, significa que la mayoría de las personas con diabetes tienen niveles de glucosa elevados.

3. **Extremos de la distribución**:
    - Si la curva de **diabetes** se extiende más hacia niveles altos de glucosa, indica que hay pacientes con diabetes con niveles mucho mayores de glucosa en sangre.
4. **Posible solapamiento**:
    - Si ambas curvas se superponen demasiado, sugiere que **solo la glucosa no es suficiente** para clasificar a los pacientes, y es necesario analizar otras variables como **BMI, Insulin o Age**.

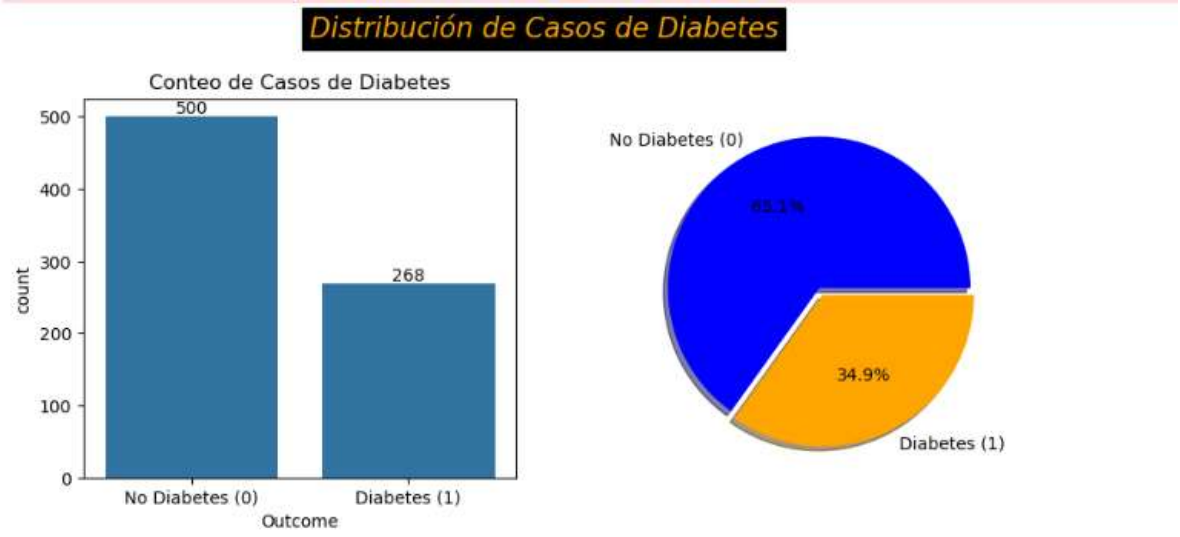
Conclusión posible (según los datos del dataset):

- Es probable que los pacientes **sin diabetes** tengan una distribución más concentrada en niveles normales de glucosa (por debajo de 120 mg/dL).
- Los pacientes **con diabetes** pueden mostrar una distribución desplazada hacia valores más altos de glucosa.

Análisis univariable

4.1. Analisis univariable

```
# Ignorar advertencias
warnings.filterwarnings("ignore", category=FutureWarning)
# Configuración del estilo del gráfico
font = {'fontsize': 16, 'fontstyle': 'italic', 'backgroundcolor': 'black', 'color': 'orange'}
# Activar gráficos en línea en Jupyter Notebook
%matplotlib inline
# Crear la figura con 2 subgráficos
fig, axes = plt.subplots(1, 2, figsize=(10, 4))

# Gráfico de barras del conteo de pacientes con y sin diabetes
sns.countplot(data=df, x='Outcome', ax=axes[0])
axes[0].set_xticklabels(['No Diabetes (0)', 'Diabetes (1)'])
axes[0].set_title('Conteo de Casos de Diabetes')

# Etiquetas en las barras
for container in axes[0].containers:
    axes[0].bar_label(container)

# Gráfico de pastel con la proporción de casos
slices = df.Outcome.value_counts().values
labels = ['No Diabetes (0)', 'Diabetes (1)']
axes[1].pie(slices, labels=labels, colors=['blue', 'orange'], shadow=True,
            explode=[0, 0.05], autopct='%1.1f%%')

# Titulo general del gráfico
plt.suptitle('Distribución de Casos de Diabetes', y=1.05, **font)

# Mostrar la visualización
plt.show()
```



La gráfica generada muestra dos representaciones visuales de la distribución de casos de diabetes en el dataset:

1. **Gráfico de barras (countplot):**
    - Muestra el número de personas sin diabetes (Outcome = 0) y con diabetes (Outcome = 1).

- Si una barra es significativamente más alta que la otra, indica un desbalance de clases.
- Un dataset desbalanceado puede afectar el desempeño de modelos predictivos, ya que el modelo podría inclinarse a predecir la clase mayoritaria.

2. **Gráfico de pastel (pie chart):**
    - Representa la proporción de pacientes con y sin diabetes.
    - Si la sección de "No Diabetes" es mucho mayor que la de "Diabetes", confirma un dataset desbalanceado.
    - Una proporción muy desigual (por ejemplo, 80%-20%) sugiere la necesidad de técnicas de balanceo, como resampling (oversampling o undersampling) o ponderación de clases al entrenar un modelo.

Análisis bivariable

4.2. Analisis Bivariable

```
# Ignorar advertencias
warnings.filterwarnings("ignore", category=FutureWarning)

# Configuración del estilo del título
font = {'fontsize': 16, 'fontstyle': 'italic', 'backgroundcolor': 'black', 'color': 'orange'}

# Activar gráficos en línea en Jupyter Notebook
%matplotlib inline

# Configurar tamaño de la figura
plt.figure(figsize=(10, 6))

# Generar el heatmap de correlaciones
sns.heatmap(df.corr(), cmap='Reds', annot=True, fmt=".2f", linewidths=0.5)

# Título del gráfico
plt.title('Mapa de Calor: Correlación entre Variables del Dataset de Diabetes', pad=15, **font)

# Mostrar el gráfico
plt.show()
```

El heatmap de correlación muestra cómo se relacionan las variables del dataset de diabetes entre sí.

- Valores cercanos a 1 (rojo oscuro) indican una fuerte correlación positiva.
- Valores cercanos a -1 indican una fuerte correlación negativa.
- Valores cercanos a 0 indican que no hay relación significativa.

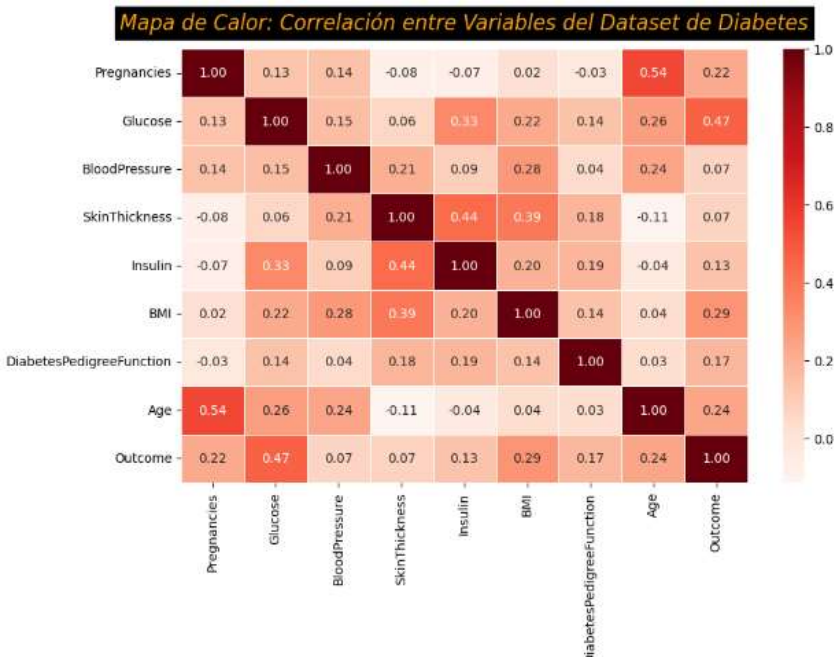


Figura 2 heatmap de correlación

Conclusión El algoritmo Naïve Bayes es una herramienta eficiente para clasificación de datos categóricos y numéricos, demostrando su utilidad en aplicaciones como diagnóstico médico. Su aplicación en la predicción de diabetes proporcionó resultados aceptables, pero con margen de mejora mediante optimización de datos y técnicas avanzadas.

Link del Github: https://github.com/Jair-Artreaga/Analisis-Datos-Naives.git