**UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO**

**FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA**

**ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS**


**PROYECTO:**

---

# "IMPLEMENTACIÓN DEL MÉTODO DE NAIVE BAYES EN PYSPARK"

---

**DOCENTE:** CARLOS FERNANDO MONTOYA CUBAS

**ALUMNOS:**

- Carcausto Mamani, Elizon Frank 170427
- Deza Condori, Rosmel Uriel 171058
- Mallqui Apaza, Nadiabeth Diana 171063
- Quispe Leon,Widmar Raul 171259
- Sullca Peralta, Melanie Indira 171070

**CUSCO – PERÚ**
**2021-II**

# I. Introducción

En el presente informe se detalla de mejor manera los conceptos incluidos en el trabajo desarrollado en el notebook de google collaboratory, presentado y expuesto en el curso presente. El trabajo desarrollado tiene como objetivo predecir si un paciente tiene o no diabetes, basándose en cierta información como es el nivel de glucosa, insulina, número de embarazos, edad y entre otros datos más los cuales se detallaran más adelante.

# II. Desarrollo

## 1. Data

En este proyecto se utilizó un dataset cuya información proviene originalmente del Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales. Extraido de Kaggle https://www.kaggle.com/mathchi/diabetes-data-set.

```
Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age,Outcome
6,148,72,35,0,33.6,0.627,50,1
1,85,66,29,0,26.6,0.351,31,0
8,183,64,0,0,23.3,0.672,32,1
1,89,66,23,94,28.1,0.167,21,0
0,137,40,35,168,43.1,2.288,33,1
5,116,74,0,0,25.6,0.201,30,0
3,78,50,32,88,31,0.248,26,1
10,115,0,0,0,35.3,0.134,29,0
2,197,70,45,543,30.5,0.158,53,1
8,125,96,0,0,0,0.232,54,1
```

**Figura 1.** Data diabetes.csv

**Contenido del dataset:**

Número de Instancias: 768
Número de atributos: 8 más clase(outcome)
Para cada atributo: (todos con valores numéricos)

- ***Pregnacies:*** Número de veces embarazadas
- ***Glucose :*** Concentración de glucosa en plasma a las 2 horas en una prueba de tolerancia oral a la glucosa.
- ***BloodPressure:*** Presión arterial diastólica (mmHg)
- ***SkinThickness:*** Grosor del pliegue cutáneo del tríceps (mm)
- ***Insulin:*** Insulina sérica de 2 horas (mu U/ml)
- ***BMI:*** Índice de masa corporal (peso en kg/(altura en m2))
- ***DiabetesPedigreeFunction:*** Función de pedigrí de diabetes
- ***Age:*** Edad (años)
- ***Outcome:*** Variable de clase (0 o 1 positivo en la prueba de diabetes)

El dataset muestra conocimiento de con qué niveles de glucosa, insulina, presión sanguínea y entre otras, se determina si una persona tiene diabetes o no.

## 2. Preprocesamiento

Para proporcionar los datos adecuados a la técnicas de machine learning, es necesario un pre procesamiento, que consiste en:

- Eliminación manual de atributos
- Integración de datos
- Muestreo de datos
- Limpieza de datos
- Transformación de datos

- Reducción de dimensionalidad

El dataset utilizado es un data que no necesita transformaciones ya que el los atributos son numéricos es por ello que solo se hará un análisis de los datos y una reducción de dimensionalidad mediante matriz de correlación.

**a. Lectura de datos y conversión**

```
#Leer archivo CSV y luego convertir a un Dataframe
df = spark.read.csv("diabetes.csv", header=True, inferSchema=True)
#Mostrar 5 instancias del data
df.show(5)
```

```
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|DiabetesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|          6|    148|           72|           35|      0|33.6|                   0.627| 50|      1|
|          1|     85|           66|           29|      0|26.6|                   0.351| 31|      0|
|          8|    183|           64|            0|      0|23.3|                   0.672| 32|      1|
|          1|     89|           66|           23|     94|28.1|                   0.167| 21|      0|
|          0|    137|           40|           35|    168|43.1|                   2.288| 33|      1|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
only showing top 5 rows
```

**Figura 2.** Muestra de 5 instancias del dataset en dataframe

**b. Observamos el tipo de datos**

```
#Acceder al tipo de cada entidad, usando printSchema ()
df.printSchema()
```

```
root
 |-- Pregnancies: integer (nullable = true)
 |-- Glucose: integer (nullable = true)
 |-- BloodPressure: integer (nullable = true)
 |-- SkinThickness: integer (nullable = true)
 |-- Insulin: integer (nullable = true)
 |-- BMI: double (nullable = true)
 |-- DiabetesPedigreeFunction: double (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Outcome: integer (nullable = true)
```

**Figura 3.** Tipo de los datos

En la imagen se puede visualizar y tener certeza que nuestros datos son numéricos

**c. Valores faltantes en el data**

```
from pyspark.sql.functions import isnan, when, count, col
### Obtener el recuento de nan o valores faltantes en pyspark
df.select([count(when(isnan(c),c)).alias(c) for c in df.columns]).toPandas()
```

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction | Age |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figura 4 .** Tabla con cantidad de datos nulos o vacíos

En la imagen se puede visualizar y tener certeza que esta todo completo

**d. Agrupación de valores de salida**

Tenemos del tipo “1” 268 personas que tienen diabetes

```
#Agrupar los valores de salida, contar y mostrar
df.groupby('Outcome').count().show()
```

```
+-------+-----+
|Outcome|count|
+-------+-----+
|      1|  268|
|      0|  500|
+-------+-----+
```

**Figura 5.** Cantidad datos salida

**e. Información estadística**

Esta información incluye el número de muestras, el valor medio, la desviación estándar, el valor mínimo, máximo, la mediana

```
#Devolver información estadística de los datos del dataframe
#Esta información incluye el número de muestras, el valor medio, la desviación estándar, el valor mínimo, máximo, la mediana
df.describe().toPandas()
```

| | summary | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI |
|---|---|---|---|---|---|---|---|
| 0 | count | 768 | 768 | 768 | 768 | 768 | 768 |
| 1 | mean | 3.8450520833333335 | 120.89453125 | 69.10546875 | 20.536458333333332 | 79.79947916666667 | 31.992578124999977 |
| 2 | stddev | 3.36957806269887 | 31.97261819513622 | 19.355807170644777 | 15.952217567727642 | 115.24400235133803 | 7.884160320375441 |
| 3 | min | 0 | 0 | 0 | 0 | 0 | 0.0 |
| 4 | max | 17 | 199 | 122 | 99 | 846 | 67.1 |

**Figura 6.** Tabla estadística

**f. Distribución de características**

```
#Distribucion de caracteristicas
fig = plt.figure(figsize=(25, 15))
st = fig.suptitle("Distribution of Features", fontsize=50, verticala
for col, num in zip(df.toPandas().describe().columns, range(1,11)):
  ax = fig.add_subplot(3,4, num)
  ax.hist(df.toPandas()[col])
  plt.grid(False)
  plt.xticks(rotation=45, fontsize=20)
  plt.yticks(fontsize=15)
  plt.title(col.upper(), fontsize=20)

plt.tight_layout()
st.set_y(0.95)
fig.subplots_adjust(top=0.85, hspace=0.4)
plt.show()
```

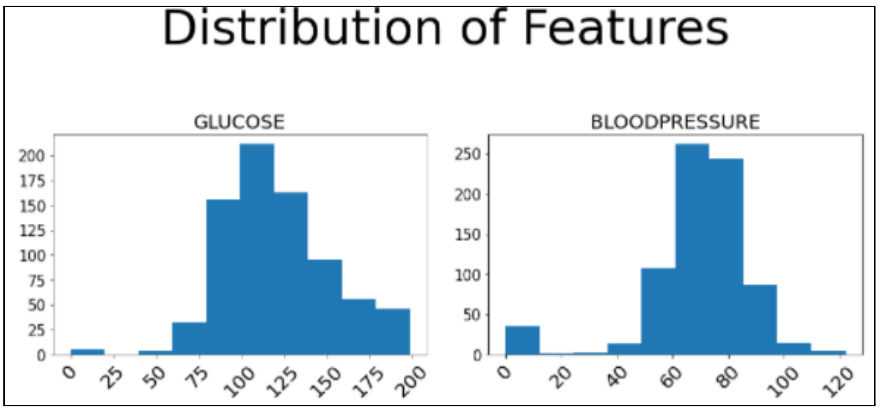


**Figura 7.** Gráfica de distribución

**g. Reducción de dimensionalidad-Matriz de correlación**

Muchos conjuntos de datos presentan un número elevado de atributos, que en muchos de los casos no son necesarios para la tarea de predicción.
La matriz de correlación es una importante métrica de análisis de datos que se calcula para resumir los datos a fin de comprender la relación entre las diversas variables y tomar decisiones en consecuencia.

***Coeficiente de correlación Pearson:*** Número que denota la fuerza de la relación entre dos variables.

```
#Matriz de correlacion con coeficiente de pearson
#Para poder hacer comparaciones se estandariza la covarianza, generando
corr_mat = Statistics.corr(features, method="pearson")
corr_df = pd.DataFrame(corr_mat)
corr_df.index = col_names
corr_df.columns = col_names
round(corr_df, 2)
```

| | Pregnancies | Glucose | BloodPressure | SkinThickness |
|---|---|---|---|---|
| **Pregnancies** | 1.00 | 0.13 | 0.14 | -0.08 |
| **Glucose** | 0.13 | 1.00 | 0.15 | 0.06 |
| **BloodPressure** | 0.14 | 0.15 | 1.00 | 0.21 |
| **SkinThickness** | -0.08 | 0.06 | 0.21 | 1.00 |
| **Insulin** | -0.07 | 0.33 | 0.09 | 0.44 |
| **BMI** | 0.02 | 0.22 | 0.28 | 0.39 |
| **DiabetesPedigreeFunction** | -0.03 | 0.14 | 0.04 | 0.18 |
| **Age** | 0.54 | 0.26 | 0.24 | -0.11 |
| **Outcome** | 0.22 | 0.47 | 0.07 | 0.07 |

**Figura 8.** Coeficiente de correlación Pearson

***Gráfico:*** Para graficar la matriz, usaremos una biblioteca de visualización popular llamada seaborn.

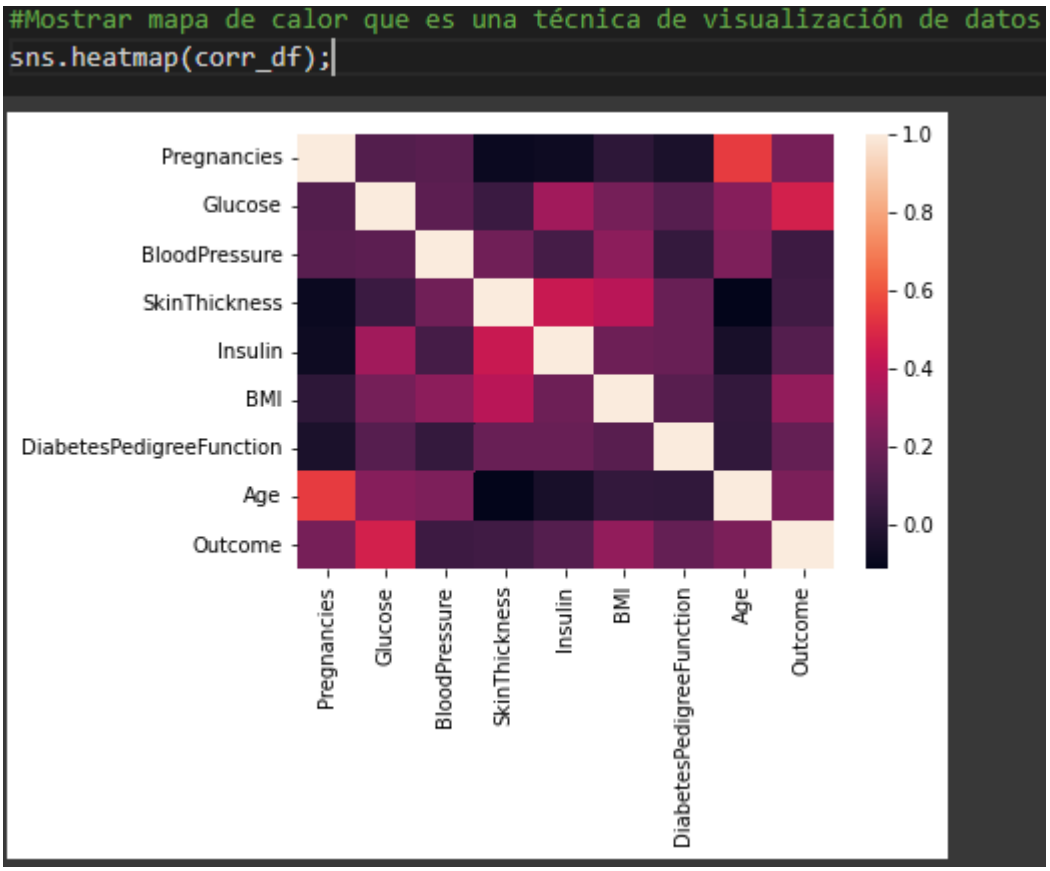


**Figura 9.**Gráfico Matriz de correlación

Hay que tener en cuenta los siguientes puntos con respecto a la matriz de correlación:

- Cada celda de la cuadrícula representa el valor del coeficiente de correlación entre dos variables.
- El valor en la posición (a, b) representa el coeficiente de correlación entre los elementos de la fila a y la columna b. Será igual al valor en la posición (b, a)
- Es una matriz cuadrada – cada fila representa una variable, y todas las columnas representan las mismas variables que las filas, de ahí el número de filas = número de columnas.
- Es una matriz simétrica – esto tiene sentido porque la correlación entre a,b será la misma que la de b,a.
- Todos los elementos diagonales son 1. Dado que los elementos diagonales representan la correlación de cada variable consigo misma, siempre será igual a 1.
- Los marcadores de los ejes denotan el rasgo que cada uno de ellos representa.
- Un valor positivo grande (cercano a 1,0) indica una fuerte correlación positiva, es decir, si el valor de una de las variables aumenta, el valor de la otra variable aumenta también.
- Un valor negativo grande (cercano a -1,0) indica una fuerte correlación negativa, es decir, que el valor de una de las variables disminuye al aumentar el de la otra y viceversa.
- Un valor cercano a 0 (tanto positivo como negativo) indica la ausencia

de cualquier correlación entre las dos variables, y por lo tanto esas variables son independientes entre sí.

Reducimos los atributos eligiendo los 5 valores con más correlación a la variable glucosa, variable escogida por ser uno de los indicadores de mayor trascendencia para determinar si una persona tiene diabetes.

```
#Variables variables con correlacion alta con Glucose
import numpy as np
k = 5
cols = corr_df.nlargest(k, 'Glucose')['Glucose'].index
cm = np.corrcoef(corr_df[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True,
fmt='.2f', annot_kws={'size': 10}, yticklabels=cols.values,
xticklabels=cols.values)
plt.show()
```

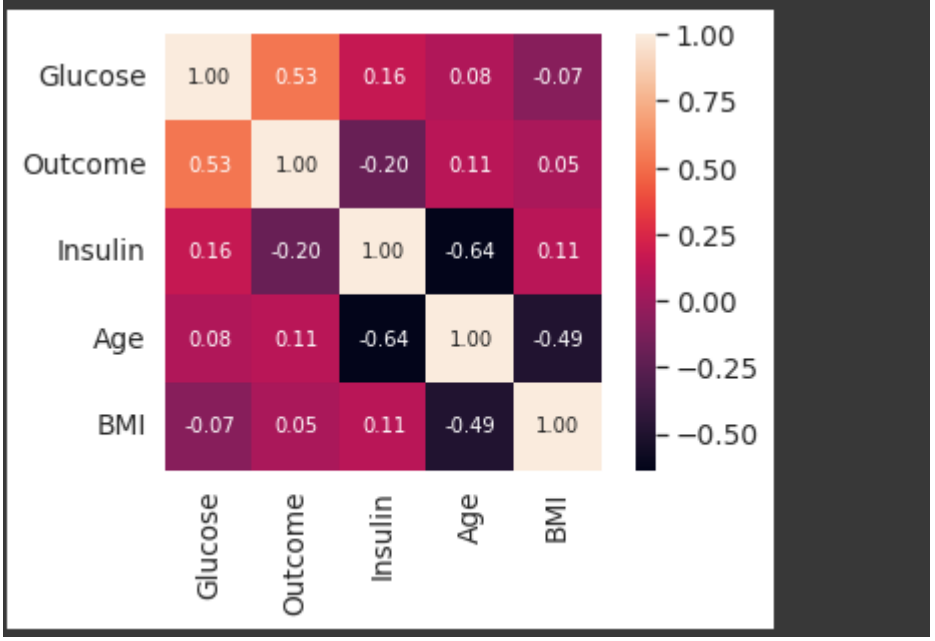


**Figura 10.** Matriz de correlación reducida

## 3. Naive Bayes

- La estructura de una red bayesiana puede utilizarse para construir clasificadores, utilizando el nodo raíz para la variable de salida o las clases.

-El proceso de clasificación se realiza usando el teorema de Bayes, para asignar a un objeto u observación la clase con mayor probabilidad.

-En el problema de clasificación, se tiene la variable de salida o de clase (C), y un conjunto de variables predictoras o atributos {A1 , A2 ,...,An }. Para éstas variables se obtiene una red bayesiana B, que define una distribución de probabilidad conjunta P (C, A1, A2, ..., An). Entonces se puede utilizar la red bayesiana resultante para dado los valores de un conjunto de atributos {a1 ,a2 ,...,an }, el clasificador basado en B retorna la etiqueta c que maximice la probabilidad a posteriori P(C = c |a1, a2, ..., an),

la clase c toma “m” posibles valores c1 , c2, ...,

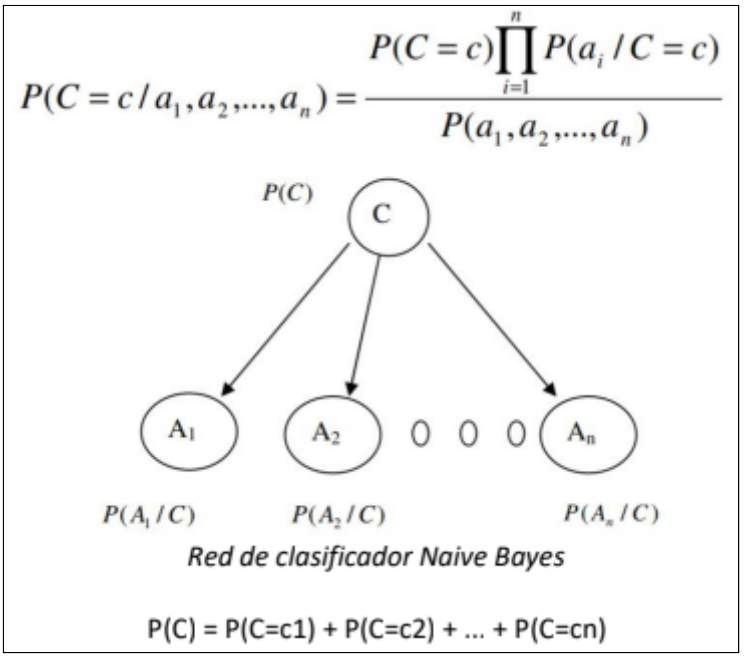


**Figura 11.** Red clasificador Naive Bayes

### 3.1 Algoritmo Naive Bayes

1. Convertir el conjunto de datos en una tabla de frecuencias.
2. Crear una tabla de probabilidad calculando las correspondientes a que ocurran los diversos eventos.
3. La ecuación Naive Bayes se usa para calcular la probabilidad posterior de cada clase**.**

**a. Cálculo de la frecuencia de la clase**

```
# Colocamos un 1 a lado de cada clase y las sumamos
frec = rdd.mapValues(lambda x: 1).reduceByKey(lambda x, y: x + y).sortByKey()
frec.collect()[:10]
```

```
[(0, 500), (1, 268)]
```

**Figura 12.** Frecuencia de las clases

**b. Cálculo de probabilidad a posteriori (predicción de nuevos casos)**
Utilizando tablas de probabilidad, media, desviación estándar

```
Para hallar la Media seguimos dos pasos
1. Sumar los valores de cada columna agrupandas por clase
2. Divimos ese resultado por el nro total de datos
'''
sumTmp = rdd.reduceByKey(lambda x, y: (x[0]+y[0], x[1]+y[1], x[2]+y[2], x[3]+y[3]) )
means = sumTmp.mapValues(lambda x: (x[0]/n, x[1]/n, x[2]/n, x[3]/n)).sortByKey()
means.collect()[:10]
```

```
[(0, (71.6015625, 44.786458333333336, 20.305989583333332, 19.729296874999974)),
 (1,
  (49.29296875, 35.013020833333336, 12.934895833333334, 12.263281249999983))]
```

**Figura 13.** Resultado media

```
def squareDistanceToMean(x):
  label = 0 if x[0] == 0 else 1
  #mean = means.collect()[label]
  mean = meansList[label]
  col0 = abs(x[1][0] - mean[0])**2
  col1 = abs(x[1][1] - mean[1])**2
  col2 = abs(x[1][2] - mean[2])**2
  col3 = abs(x[1][2] - mean[3])**2
  return (label, (col0, col1, col2, col3))
```

```
'''
Para hallar la desviacion estandar:
1. Calculamos el cuadrado de la distancia a la media de cada valor
2. Sumamos las distancias de cada columna agrupandolas por clase
3. Sacamos la raiz cuadrada de la suma de distancias sobre el nro total de datos
'''
distanceToMean = rdd.map(squareDistanceToMean)
sumDistances = distanceToMean.reduceByKey(lambda x, y: (x[0]+y[0], x[1]+y[1], x[2]+y[2], x[3]+y[3])).sortByKey()
devStdr = sumDistances.mapValues(lambda x: ((x[0]/n)**0.5, (x[1]/n)**0.5, (x[2]/n)**0.5, (x[3]/n)**0.5))
devStdr.collect()
```

```
[(0,
  (37.455703568075734,
   82.01190353891874,
   12.867604640579708,
   13.189563822497119)),
 (1,
  (57.49749426081236,
   90.42169765083963,
   15.653930703214288,
   16.016069814369644))]
```

**Figura 13.** Cálculo desviaciones estándar

### c. Cálculo de probabilidad a posteriori (predicción de nuevos casos)

$$P(A_i / c) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(X-\mu)^2}{2\sigma^2}\right)$$

**Figura 13.** Formula para hallar predicción

```
from math import pi, exp, sqrt
'''
Encontramos la probabilidad de cada dato
usando la distribucion normal:

'''
def getProb(x, label, col):
  mean = meansList[label][col]
  dev = devStdrList[label][col]
  var = dev**2
  factor1 = (1) / ((sqrt(2*pi)) * dev)
  factor2 = exp(-( ((x-mean)**2) / (2*var)))
  return factor1 * factor2
```

**Figura 13.**Módulo probabilidad

```python
def getProbsApriori(dat):
  # Encontramos las probabilidades de cada clase
  probClassTmp = [tuple(frecList)]
  probClass = list(map(lambda x: (x[0]/n, x[1]/n), probClassTmp))

  # Calculamos las probabilidades de los datos que queremos
  # predecir usando el modulo q evalua la distribucion normal
  probData = []
  for i in range(len(dat)):
    prob = (getProb(dat[i], 0, i), getProb(dat[i], 1, i))
    probData.append(prob)

  # unismo las dos tablas
  probApriori = probClass + probData
  return probApriori


'''
En base a las probabilidades apriori
calculamos la probabilidad a posteriori
para poder generar una prediccion
'''
def getProbPosteriori(probApriori):
  sc = spark.sparkContext
  predictRDD = sc.parallelize(probApriori)
  # Multiplicamos las probabilidades para cada clase
  f = predictRDD.reduce(lambda x, y: (x[0]*y[0], x[1]*y[1]))

  # calculamos la probabilidad a posteriori
  probPosteriori = tuple(map(lambda x: round((x/sum(f))*100, 2), f))
  return probPosteriori
```

**Figura 14.**Módulo probabilidad a priori y posteriori

## 4. Resultados

Formato datos a ingresar para predicción, estos tienen que ser valores numéricos

**[ glucosa, insulina, edad, Indice de masa corporal(IMC)]**

```python
datos1 = [100, 94, 40, 30]
predict1 = getProbPosteriori(probApriori1)
print('Tiene diabetes: ', datos1)
print('NO :', predict1[0], '%')
print('SI :', predict1[1], '%')
```

```
Tiene diabetes:  [100, 94, 40, 30]
NO : 90.95 %
SI : 9.05 %
```

**Figura 15.**Ejemplo predicción 1

Finalmente para elegir la clase a la que pertenece, elegimos la probabilidad de mayor valor de la dos anteriores:

Se observa en el segundo resultado un valor al 90.95% de que la persona no tiene diabetes

```
datos4 = [200, 168, 20, 15]
predict4 = getProbPosteriori(getProbsApriori(datos4))
print('Tiene diabetes: ', datos2)
print('NO :', predict4[0], '%')
print('SI :', predict4[1], '%')
```

```
Tiene diabetes:  [180, 86, 60, 25.5]
NO : 28.99 %
SI : 71.01 %
```

**Figura 16.**Ejemplo predicción 2

Se observa que el resultado nos devuelve un valor al 71.01% de que la persona si tiene diabetes

## III. Conclusión

Del presente trabajo se concluyen los siguientes puntos:

- La herramienta Pyspark funciona bien para el procesamiento de datos
- La matriz de correlación es útil para descartar variables de poca importancia
- El algoritmo Naive Bayes es un método corto y rápido de hacer predicciones haciendo uso de valores estadísticos, sin la necesidad de hacer uso de librerías o realizar entrenamiento de datos.
- Se pudo identificar que realmente el valor de la glucosa, es un indicador con mayor relevancia al momento de predecir si una persona tiene diabetes ya que si este es mayor determina positivo.
- Fue una buena elección el atributo Glucose para la reducción de dimensionalidad.

## IV. Referencias


https://experienceleague.adobe.com/docs/data-workbench/using/client/analysis-visualizations/correlation-analysis/c-correlation-analysis.html?lang=es

https://likegeeks.com/es/matrix-correlacion-python/

https://drive.google.com/file/d/17g_qRWQHxlo05TxSBtObCctJ2ztzTm_g/view

https://drive.google.com/file/d/1aEI-ynlZwTu5ujHHy7dSXxJLz_xdtpNj/view

https://www.kaggle.com/mathchi/diabetes-data-set