En este proyecto se mostrará un análisis exhaustivo sobre las principales características que califican a pacientes con diabetes. El análisis se realizó con dos datasets obtenidos que contienen datos y atributos de pacientes positivos a diabetes, el objetivo es hallar y mostrar con estos datos las características que exponen y/o resaltan como importante en personas con diabetes.

Los métodos de desarrollo están fundamentados y sustentados en base al Análisis de datos y Machine Learning usando las siguientes herramientas y bibliotecas; Python, Numpy, Pandas, Scipy, Matplotlib, Árboles de decisión, Entrenamiento y Prueba, K-means y Clustering.

Dataset 1

Objetivo: El objetivo de esta primera parte será exponer un listado ordenado de cada característica con respecto a su nivel de importancia o relevancia después de hacer una prueba de predicción, realizado con el método de Árboles de decisión.

Variables:

Valores de entrada (Características):

- Age
- Gender
- Polyuria
- Polydipsia
- sudden weight loss
- weakness
- Polyphagia
- Genital thrush
- visual blurring
- Itching
- Irritability
- delayed healing
- partial paresis
- muscle stiffness
- Alopecia
- Obesity

Valor de salida:

- Class = Clase

Se importa el archivo **'diabetes_data_upload.csv'** y se guarda los datos en la variable **data1.**

```
#se asigna los valores del archivo dataset1.csv a data1
data1 = pd.read_csv('diabetes_data_upload.csv')
data1.head(5)
```

	Age	Gender	Polyuria	Polydipsia	sudden weight loss	weakness	Polyphagia	Genital thrush	visual blurring	Itching	Irritability	delayed healing	partial paresis	muscle stiffness	Alopecia	Obesity	class
0	40	Male	No	Yes	No	Yes	No	No	No	Yes	No	Yes	No	Yes	Yes	Yes	Positive
1	58	Male	No	No	No	Yes	No	No	Yes	No	No	No	Yes	No	Yes	No	Positive
2	41	Male	Yes	No	No	Yes	Yes	No	No	Yes	No	Yes	No	Yes	Yes	No	Positive
3	45	Male	No	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes	No	No	No	No	Positive
4	60	Male	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Positive

Información de los datos:

```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 520 entries, 0 to 519
Data columns (total 17 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Age                 520 non-null    int64 
 1   Gender              520 non-null    object
 2   Polyuria            520 non-null    object
 3   Polydipsia          520 non-null    object
 4   sudden weight loss  520 non-null    object
 5   weakness            520 non-null    object
 6   Polyphagia          520 non-null    object
 7   Genital thrush      520 non-null    object
 8   visual blurring     520 non-null    object
 9   Itching             520 non-null    object
 10  Irritability        520 non-null    object
 11  delayed healing     520 non-null    object
 12  partial paresis     520 non-null    object
 13  muscle stiffness    520 non-null    object
 14  Alopecia            520 non-null    object
 15  Obesity             520 non-null    object
 16  class               520 non-null    object
dtypes: int64(1), object(16)
memory usage: 69.2+ KB
```

Se observa que hay un total de 520 registros en las 17 columnas.

```
pd.value_counts(data1['clase'])
```

```
Positive    320
Negative    200
Name: clase, dtype: int64
```

Y 2 tipos de clase: Positive(Yes) y Negative(No).

Descripción general de los datos:

```
data1.describe()
```

	Age
count	520.000000
mean	48.028846
std	12.151466
min	16.000000
25%	39.000000
50%	47.500000
75%	57.000000
max	90.000000

(Categorizado con valores '0','1')

```
data1.describe()
```

	Age	Gender	Polyuria	Polydipsia	sudden	weakness	Polyphagia	Genital	visual	Itching	Irritability
count	520.000000	520.000000	520.000000	520.000000	520.000000	520.000000	520.000000	520.000000	520.000000	520.000000	520.000000
mean	48.028846	0.630769	0.496154	0.448077	0.417308	0.586538	0.455769	0.223077	0.448077	0.486538	0.242308
std	12.151466	0.483061	0.500467	0.497776	0.493589	0.492928	0.498519	0.416710	0.497776	0.500300	0.428892
min	16.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	39.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	47.500000	1.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	57.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000	1.000000	0.000000
max	90.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Tipo de datos:

```
data1.dtypes
```

```
Age                  int64
Gender              object
Polyuria            object
Polydipsia          object
sudden weight loss  object
weakness            object
Polyphagia          object
Genital thrush      object
visual blurring     object
Itching             object
Irritability        object
delayed healing     object
partial paresis     object
muscle stiffness    object
Alopecia            object
Obesity             object
class               object
dtype: object
```

Se categoriza los datos clasificando sus valores a valores de tipo int para luego trabajar con ellos en estadística, entrenamiento y en el árbol de decisiones.

Se cambia los nombres de las columnas:

```
data1.columns = ['Age', 'Gender', 'Polyuria', 'Polydipsia', 'sudden', 'weakness', 'Polyphagia', 'Genital', 'visual',
                 'Itching', 'Irritability', 'delayed', 'partial', 'muscle', 'Alopecia', 'Obesity', 'clase']
data1.columns
```

```
Index(['Age', 'Gender', 'Polyuria', 'Polydipsia', 'sudden', 'weakness',
       'Polyphagia', 'Genital', 'visual', 'Itching', 'Irritability', 'delayed',
       'partial', 'muscle', 'Alopecia', 'Obesity', 'clase'],
      dtype='object')
```

Con la función LabelEncoder() de la librería sklearn.preprocessing y la función fit_transform para transformar los valores, se categoriza los valores que se tenía en cada columna a valores enteros como '0' y '1'

```
encoder=LabelEncoder()
data1['Gender']=encoder.fit_transform(data1.Gender.values)
data1['Polyuria']=encoder.fit_transform(data1.Polyuria.values)
data1['Polydipsia']=encoder.fit_transform(data1.Polydipsia.values)
data1['sudden']=encoder.fit_transform(data1.sudden.values)
data1['weakness']=encoder.fit_transform(data1.weakness.values)
data1['Polyphagia']=encoder.fit_transform(data1.Polyphagia.values)
data1['Genital']=encoder.fit_transform(data1.Genital.values)
data1['visual']=encoder.fit_transform(data1.visual.values)
data1['Itching']=encoder.fit_transform(data1.Itching.values)
data1['Irritability']=encoder.fit_transform(data1.Irritability.values)
data1['delayed']=encoder.fit_transform(data1.delayed.values)
data1['partial']=encoder.fit_transform(data1.partial.values)
data1['muscle']=encoder.fit_transform(data1.muscle.values)
data1['Alopecia']=encoder.fit_transform(data1.Alopecia.values)
data1['Obesity']=encoder.fit_transform(data1.Obesity.values)
data1['clase']=encoder.fit_transform(data1.clase.values)
data1
```

Quedando así:

	Age	Gender	Polyuria	Polydipsia	sudden	weakness	Polyphagia	Genital	visual	Itching	Irritability	delayed	partial	muscle	Alopecia	Obesity	clase
0	40	1	0	1	0	1	0	0	0	1	0	1	0	1	1	1	1
1	58	1	0	0	0	1	0	0	1	0	0	0	1	0	1	0	1
2	41	1	1	0	0	1	1	0	0	1	0	1	0	1	1	0	1
3	45	1	0	0	1	1	1	1	0	1	0	1	0	0	0	0	1
4	60	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
515	39	0	1	1	1	0	1	0	0	1	0	1	1	0	0	0	1
516	48	0	1	1	1	1	1	0	0	1	1	1	1	0	0	0	1
517	58	0	1	1	1	1	1	0	1	0	0	0	1	1	0	1	1
518	32	0	0	0	0	1	0	0	1	1	0	1	0	0	1	0	0
519	42	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

520 rows × 17 columns

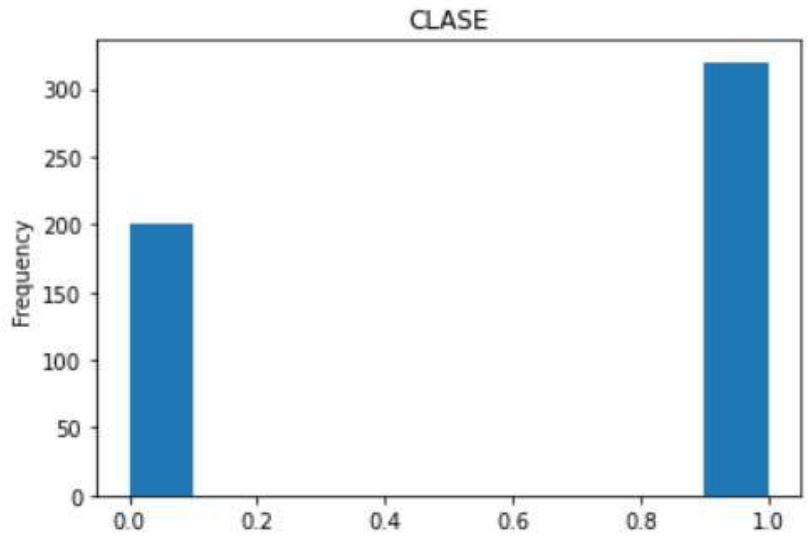
Y posteriormente se ponen los valores en un array:

```
Gender = np.array(data1.Gender)
Polyuria = np.array (data1.Polyuria)
Polydipsia = np.array(data1.Polydipsia)
sudden= np.array(data1.sudden)
weakness= np.array(data1.weakness)
Polyphagia= np.array(data1.Polyphagia)
Polyphagia= np.array(data1.Polyphagia)
Genital= np.array(data1.Genital)
visual= np.array(data1.visual)
Itching= np.array(data1.Itching)
Irritability= np.array(data1.Irritability)
delayed= np.array(data1.delayed)
partial= np.array(data1.partial)
muscle= np.array(data1.muscle)
Alopecia= np.array(data1.Alopecia)
clase= np.array(data1.clase)
Age= np.array(data1.Age)
```

Estadística de datos:

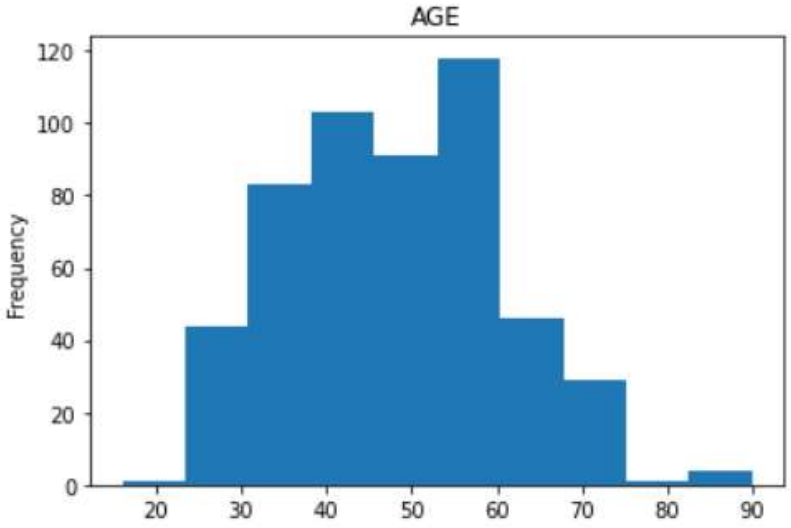
Clase:

```
data1 ['clase'].plot.hist(title='CLASE')
plt.show()
```



Edad:

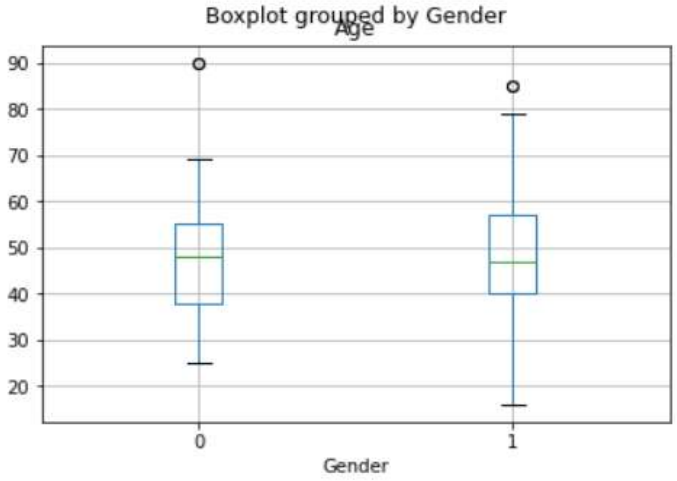
```
data1 ['Age'].plot.hist(title='AGE')
plt.show()
```



Representando de forma visual las variables utilizando el tipo de gráfico **boxplot.**

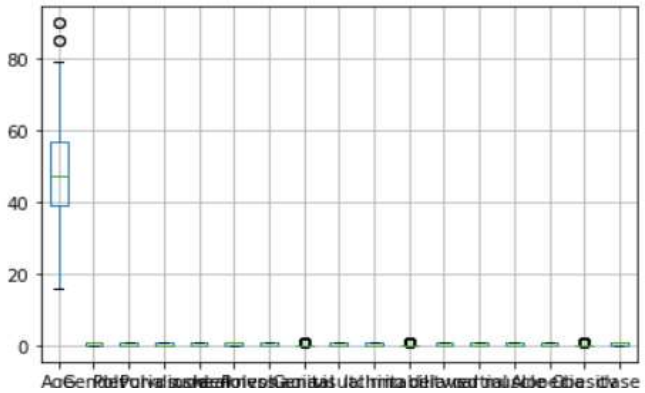
```
data1.boxplot('Age','Gender')
```

```
<AxesSubplot:title={'center':'Age'}, xlabel='Gender'>
```



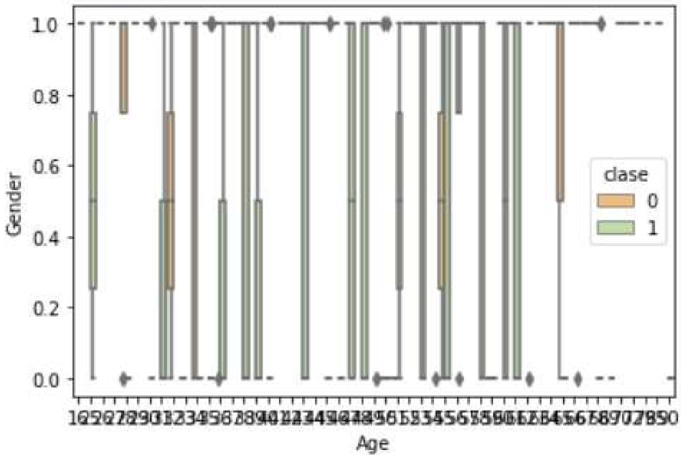

```
data1.boxplot(column=['Age','Gender','Polyuria','Polydipsia','sudden','weakness','Polyphagia','Genital',
                      'visual','Itching', 'Irritability','delayed','partial', 'muscle', 'Alopecia',
                      'Obesity', 'clase'])
```

```
<AxesSubplot:>
```



Con 3 variables: “Age”, ”Gender” y ”clase”

```
sns.boxplot(x="Age", y="Gender", hue="clase", data=data1 , palette="Spectral")
plt.show()
```



Dividiendo el conjunto de datos en **“x”** como entrada y **”y”** como salida.

Poniendo en “**x”** los valores de la columna:
'**Age**','**Gender**','**Polyuria**','**Polydipsia**','**sudden**','**weakness**','**Polyphagia**','**Genital**', '**visual**','**Itching**', '**Irritability**','**delayed**','**partial**', '**muscle**', '**Alopecia**', '**Obesity**'.

Y poniendo en “**y”** los valores de la columna '**clase**'.

```
x = data1.iloc[:,0:16].values
y = data1.iloc[:, 16].values
```

```
x
```

```
array([[40,  1,  0, ...,  1,  1,  1],
       [58,  1,  0, ...,  0,  1,  0],
       [41,  1,  1, ...,  1,  1,  0],
       ...,
       [58,  0,  1, ...,  1,  0,  1],
       [32,  0,  0, ...,  0,  1,  0],
       [42,  1,  0, ...,  0,  0,  0]], dtype=int64)
```

```
y
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

Dividiendo el conjunto de datos en **25%** (test)“test_size” **y 75%** en entrenamiento con la función **train_test_split.**

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
```

Se importa la función **DecisionTreeClassifier** y se asigna al **clasificador** con los valores:

Criterion = **entropy** y Max_depth = **6**

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', max_depth = 4, random_state = 0)
classifier.fit(x_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=4, random_state=0)
```

Se genera la matriz de confusión:

```
y_pred = classifier.predict(x_test)
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[51,  5],
       [ 4, 70]], dtype=int64)
```

Se obtiene el valor "**Accuracy**", "F1 Score" y "ROC":

```
from sklearn import metrics
print("Accuracy: ", metrics.accuracy_score(y_test, y_pred))
print("F1 Score: ", metrics.f1_score(y_test, y_pred, average = 'weighted'))
print("ROC: ", metrics.roc_auc_score(y_test, y_pred))
```

```
Accuracy:  0.9307692307692308
F1 Score:  0.9306896984749333
ROC:  0.9283301158301158
```

Accuracy = 0.9307

Con Criterion = **gini** y Max_deph = **5**

```
from sklearn import metrics
print("Accuracy: ", metrics.accuracy_score(y_test, y_pred))
print("F1 Score: ", metrics.f1_score(y_test, y_pred, average = 'weighted'))
print("ROC: ", metrics.roc_auc_score(y_test, y_pred))
```

```
Accuracy:  0.9692307692307692
F1 Score:  0.969289926607879
ROC:  0.9708011583011582
```

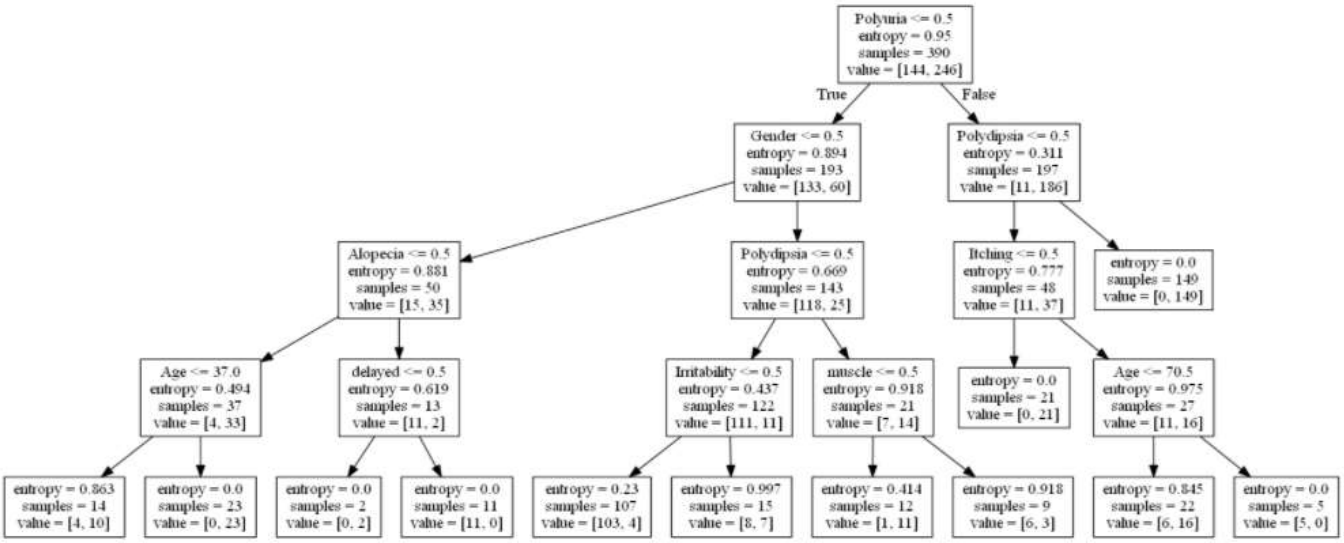
Accuracy = 0.9692

Tabla de hyperparámetros y resultado:

criterion	Max_depth	Resultado(Accuracy)
Gini	5	0.9692
Entropy	6	0.9307

Se genera el **árbol de decisiones** con los parámetros: **criterion = 'entropy'** y **max_depth = 6**

```
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```



Se realiza una prueba de predicción con los siguientes valores:

```
newdata = [[40,1,0,0,1,1,1,0,1,0,1,1,0,1,0,1],[34,1,1,0,1,1,1,1,1,1,0,1,1,0,0,0,0]]
classifier.predict(newdata)
```

```
array([0, 1])
```

Con la función **feature_importances_** se muestra un listado ordenado de cada columna con su respectiva importancia o relevancia después de realizar la predicción.

```
importancia_predictores = pd.DataFrame(
                            {'predictor': data1.drop(columns = "clase").columns,
                             'importancia': classifier.feature_importances_}
                            )
importancia_predictores.sort_values('importancia', ascending=False)
```

	predictor	importancia
2	Polyuria	0.476400
3	Polydipsia	0.163550
1	Gender	0.114619
14	Alopecia	0.061761
0	Age	0.048535
10	Irritability	0.047857
9	Itching	0.038136
11	delayed	0.028052
13	muscle	0.021090
4	sudden	0.000000
5	weakness	0.000000
6	Polyphagia	0.000000
7	Genital	0.000000
8	visual	0.000000
12	partial	0.000000
15	Obesity	0.000000

Como conclusión de la primera parte, obtuvimos que las 5 características más relevantes y comunes en pacientes con diabetes son:

1. **Polyuria** (Producción de orina de > 3 L por día)
2. **Polydipsia** (Tener mucha sed y tomar una gran cantidad de líquido)
3. **Gender** (Género, siendo hombres con mayor número de registros, 63%)
4. **Alopecia** (Pérdida anormal del cabello)
5. **Age** (Edad)

Dataset 2

Objetivo: El objetivo de esta segunda parte será demostrar la relación que tienen las características específicamente el peso y la edad con respecto a un paciente positivo a la diabetes, realizado con el método de Clusterización.

Variables:

Preg = Número de veces de embarazo.

Plas = Concentración de glucosa.

Pres = Presión arterial en mm Hg.

Skin = Grosor del pliegue de la piel en mm.

Insu = Insulina.

Mass = Índice de masa corporal.

Pedi = Función de pedigrí de diabetes.

Age = Edad.

Se importa y se muestra las primeras 5 filas de los datos del archivo "diabetes.csv"

	preg	plas	pres	skin	insu	mass	pedi	age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

Con el código **data = pd.read_csv('diabetes.csv')** para importar y **data.head(5)** para mostrar

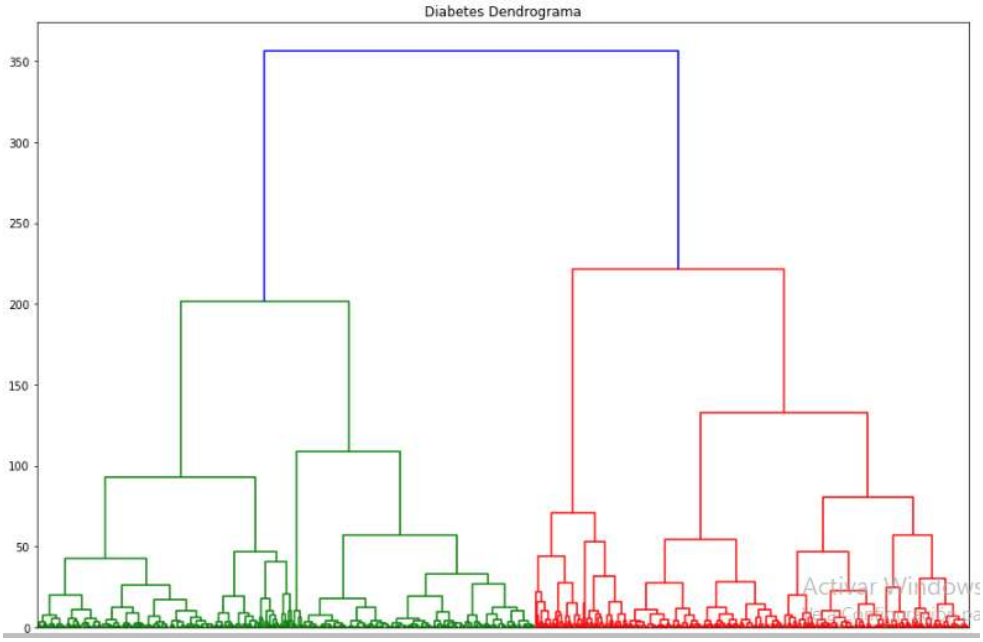
Se asigna los datos de las columnas ("mass", "pedi" y "age") en un array para luego usar los métodos para agrupar y clasificar.

Código: **data1 = data.iloc[:, 5:8].values**

Se corrobora con una muestra donde los datos de las columnas "mass","pedi","age" correspondientemente ya están en un array listos para ser evaluados.

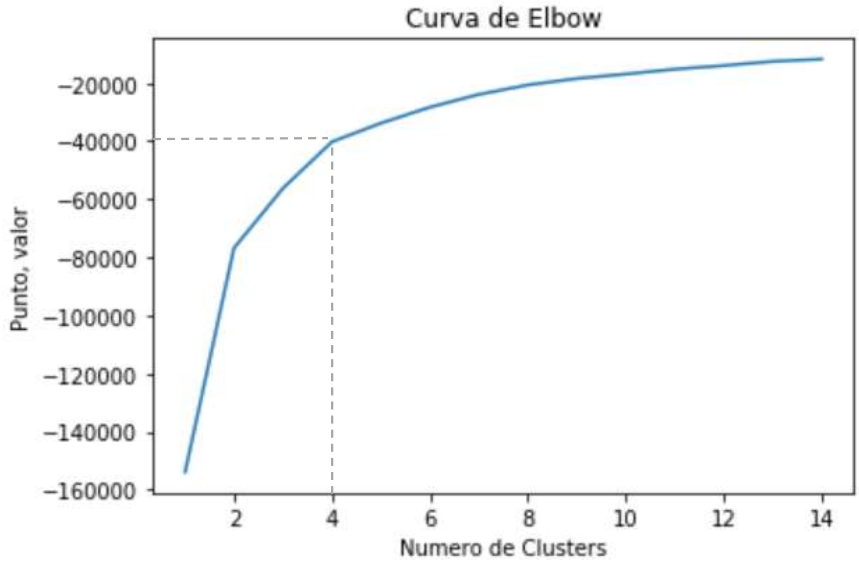
```
array([[33.6  ,  0.627, 50.   ],
       [26.6  ,  0.351, 31.   ],
       [23.3  ,  0.672, 32.   ],
       ...,
       [26.2  ,  0.245, 30.   ],
       [30.1  ,  0.349, 47.   ],
       [30.4  ,  0.315, 23.   ]])
```

Con el método jerárquico ya se puede observar y decir cuántos grupos o clusters puede haber, pero con el método Elbow es más fácil decidir el número de clusters.



Hallando el valor K con Elbow Method.

Aquí se hace el uso del método Elbow, el vértice ubicado más cerca a la curva indica el mejor número de clusters, en este caso el 4 está en la mejor posición.



Ejecutamos K-means.

K-means toma valor a cada dato clusterizando, en este caso con 4 clusters, a continuación el código y valores clústerizados por K-means:

```
from sklearn.cluster import KMeans

km = KMeans(n_clusters=4, random_state=20).fit_predict(data1)
```

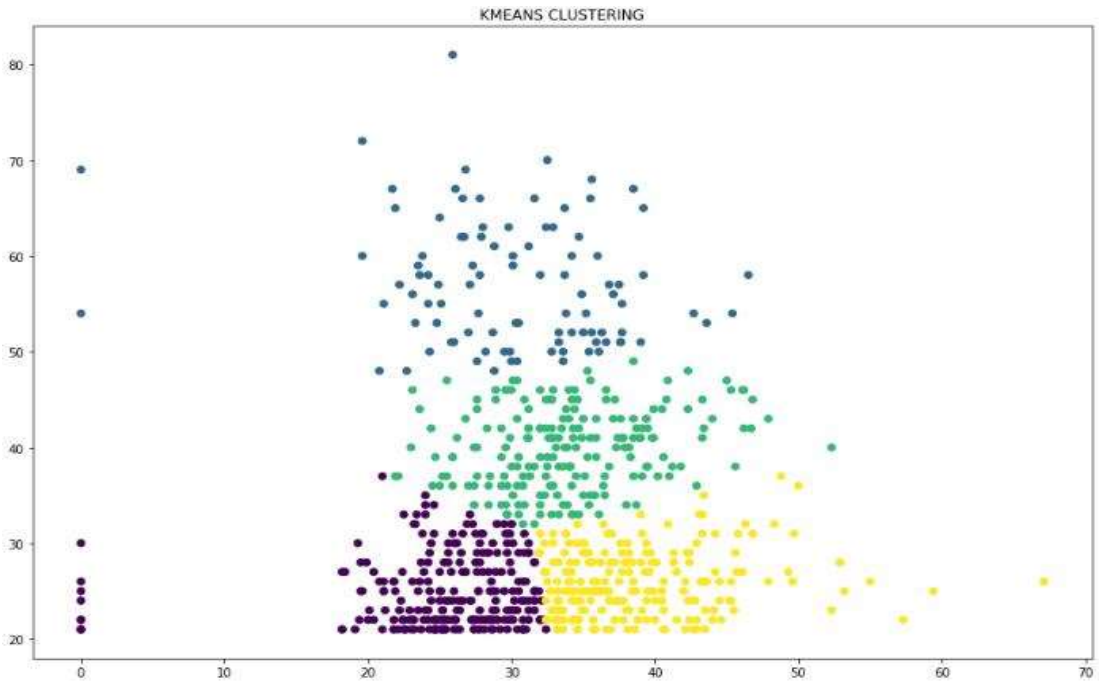
```
array([1, 0, 0, 0, 3, 0, 0, 3, 1, 1, 3, 2, 1, 1, 1, 0, 3, 0, 3, 3, 3, 1,
       2, 0, 1, 2, 2, 0, 1, 2, 1, 0, 0, 0, 2, 0, 2, 2, 3, 1, 3, 2, 1, 1,
       2, 3, 0, 0, 3, 0, 0, 0, 0, 1, 2, 0, 2, 3, 2, 3, 0, 2, 2, 0, 2, 0,
       2, 1, 0, 0, 3, 0, 2, 3, 0, 0, 2, 3, 3, 0, 0, 0, 2, 0, 3, 3, 2, 3,
       2, 0, 0, 2, 2, 1, 0, 2, 0, 0, 0, 3, 3, 0, 0, 0, 3, 0, 0, 2, 3, 3,
       3, 2, 0, 3, 2, 1, 2, 3, 0, 0, 3, 3, 3, 1, 3, 3, 3, 3, 2, 1, 2, 2,
       3, 2, 0, 3, 0, 0, 0, 3, 1, 2, 3, 2, 3, 0, 2, 2, 1, 0, 3, 2, 2, 3,
       2, 3, 0, 0, 0, 2, 2, 2, 3, 0, 2, 2, 3, 2, 3, 0, 2, 3, 0, 3, 2, 2,
       2, 3, 2, 2, 0, 3, 0, 0, 2, 2, 1, 2, 0, 0, 0, 2, 2, 2, 2, 3, 0, 0,
       3, 0, 0, 3, 0, 0, 1, 0, 1, 1, 3, 2, 0, 3, 1, 3, 2, 2, 3, 0, 0, 2,
       3, 1, 2, 1, 0, 3, 3, 3, 3, 3, 3, 2, 0, 3, 0, 3, 1, 3, 2, 0, 0, 3,
       0, 0, 3, 1, 2, 3, 2, 0, 2, 0, 0, 3, 2, 3, 0, 0, 0, 1, 2, 0, 0, 1,
       2, 2, 3, 3, 0, 0, 2, 0, 2, 3, 1, 3, 0, 0, 1, 0, 3, 2, 2, 2, 1, 1,
       2, 3, 0, 2, 3, 3, 3, 3, 1, 3, 0, 0, 2, 1, 3, 0, 2, 3, 0, 3, 2, 0,
       0, 3, 2, 3, 0, 0, 2, 3, 0, 0, 3, 1, 0, 0, 2, 2, 3, 0, 3, 2, 3, 2,
       2, 3, 2, 2, 0, 3, 2, 2, 2, 0, 2, 0, 2, 1, 2, 0, 0, 0, 2, 3, 0,
       2, 0, 3, 1, 3, 2, 2, 3, 0, 1, 1, 1, 3, 3, 0, 0, 0, 2, 3, 0, 3, 3,
       3, 1, 0, 3, 3, 3, 0, 0, 0, 0, 0, 0, 2, 2, 1, 0, 2, 3, 0, 2, 3, 0,
       2, 3, 0, 3, 3, 1, 2, 2, 2, 3, 2, 0, 2, 3, 3, 3, 3, 0, 3, 3, 0, 2,
       0, 0, 3, 0, 3, 0, 2, 3, 0, 2, 3, 2, 0, 2, 0, 0, 2, 3, 2, 0, 0, 3,
       2, 0, 3, 2, 0, 3, 0, 3, 3, 0, 0, 0, 3, 1, 3, 2, 1, 0, 1, 1, 1, 0,
       2, 2, 0, 0, 0, 3, 2, 3, 3, 3, 3, 1, 0, 1, 3, 0, 2, 1, 2, 3, 0, 3,
       3, 3, 3, 1, 0, 1, 3, 2, 2, 2, 0, 1, 0, 0, 1, 2, 0, 3, 2, 2, 2, 2,
       2, 0, 0, 1, 2, 0, 1, 0, 0, 0, 1, 1, 2, 1, 0, 3, 0, 2, 0, 0, 0, 0,
       0, 0, 0, 3, 3, 0, 3, 3, 2, 1, 3, 3, 2, 3, 1, 3, 3, 2, 1, 3, 1, 2,
       0, 0, 1, 0, 3, 2, 3, 1, 2, 2, 1, 3, 3, 0, 3, 0, 3, 2, 2, 2, 2, 0,
       0, 3, 0, 3, 0, 3, 2, 1, 3, 0, 1, 2, 1, 0, 2, 0, 1, 0, 2, 3, 2, 0,
       2, 0, 2, 0, 2, 0, 0, 0, 0, 1, 2, 3, 3, 0, 3, 0, 0, 2, 2, 3, 1, 0,
       0, 0, 1, 3, 3, 0, 2, 3, 0, 3, 0, 0, 2, 0, 2, 3, 0, 0, 0, 2, 1, 0,
       3, 0, 0, 3, 1, 0, 0, 3, 0, 3, 2, 0, 0, 3, 3, 0, 3, 3, 0, 2, 1, 3,
       1, 3, 2, 2, 2, 3, 1, 2, 2, 2, 1, 0, 2, 3, 1, 0, 1, 3, 3, 0, 0, 3,
       3, 3, 1, 3, 0, 0, 0, 2, 0, 2, 3, 2, 0, 2, 0, 0, 3, 3, 3, 1, 1, 2,
       0, 3, 0, 3, 2, 3, 0, 2, 2, 0, 0, 2, 3, 1, 3, 1, 2, 3, 2, 2, 2, 2,
       3, 3, 0, 0, 2, 0, 3, 0, 1, 3, 0, 2, 3, 2, 2, 0, 0, 2, 2, 2, 3, 3,
       2, 1, 0, 3, 0, 3, 2, 2, 2, 1, 3, 1, 0, 2, 0, 1, 3, 0, 2, 0])
```

Como podemos observar el método K-means asignó a cada dato un valor de entre 0 a 4 que significa "K" clusters.

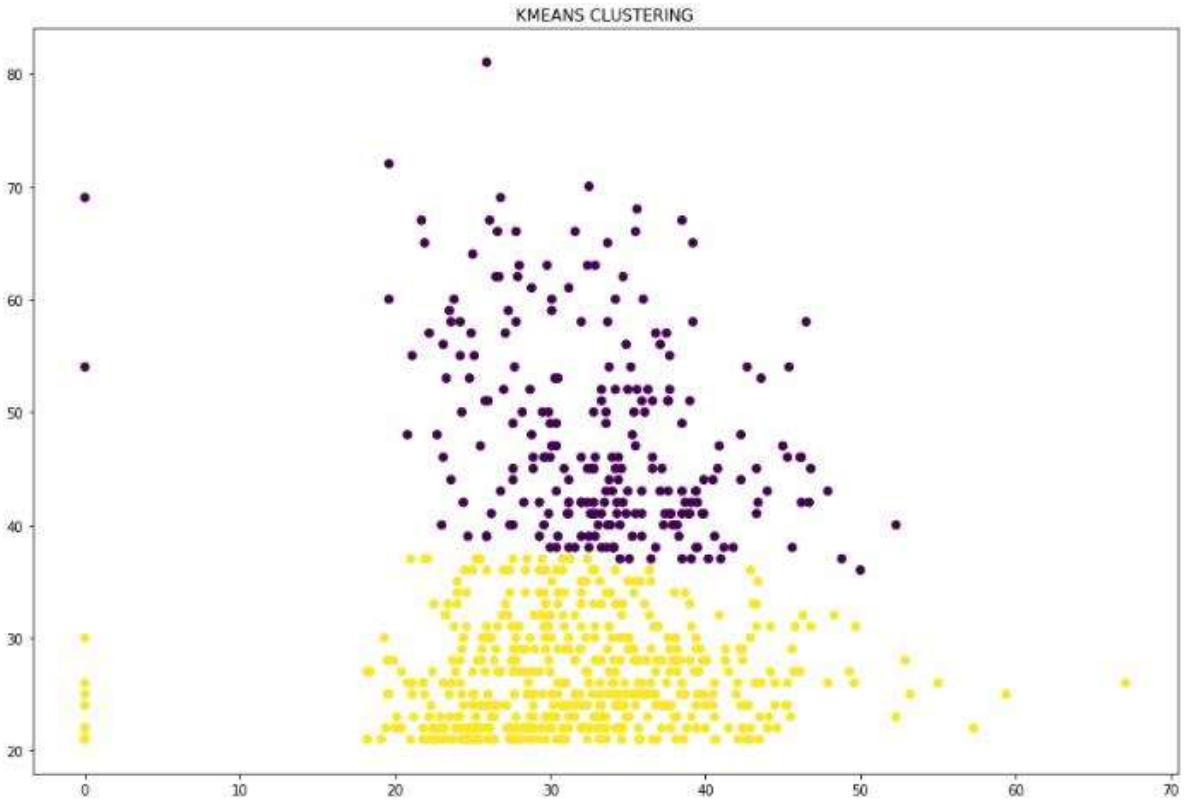
Aquí ya se muestra gráficamente la agrupación por K-means, se ven 4 colores diferentes la cual significa que se tomó k=4 (clusters), se ve claramente la agrupación dada las variablesmás primordiales.

Se mostrarán los resultados obtenido con diferentes números de cluster(k) a fin de entender porque se eligió 4.

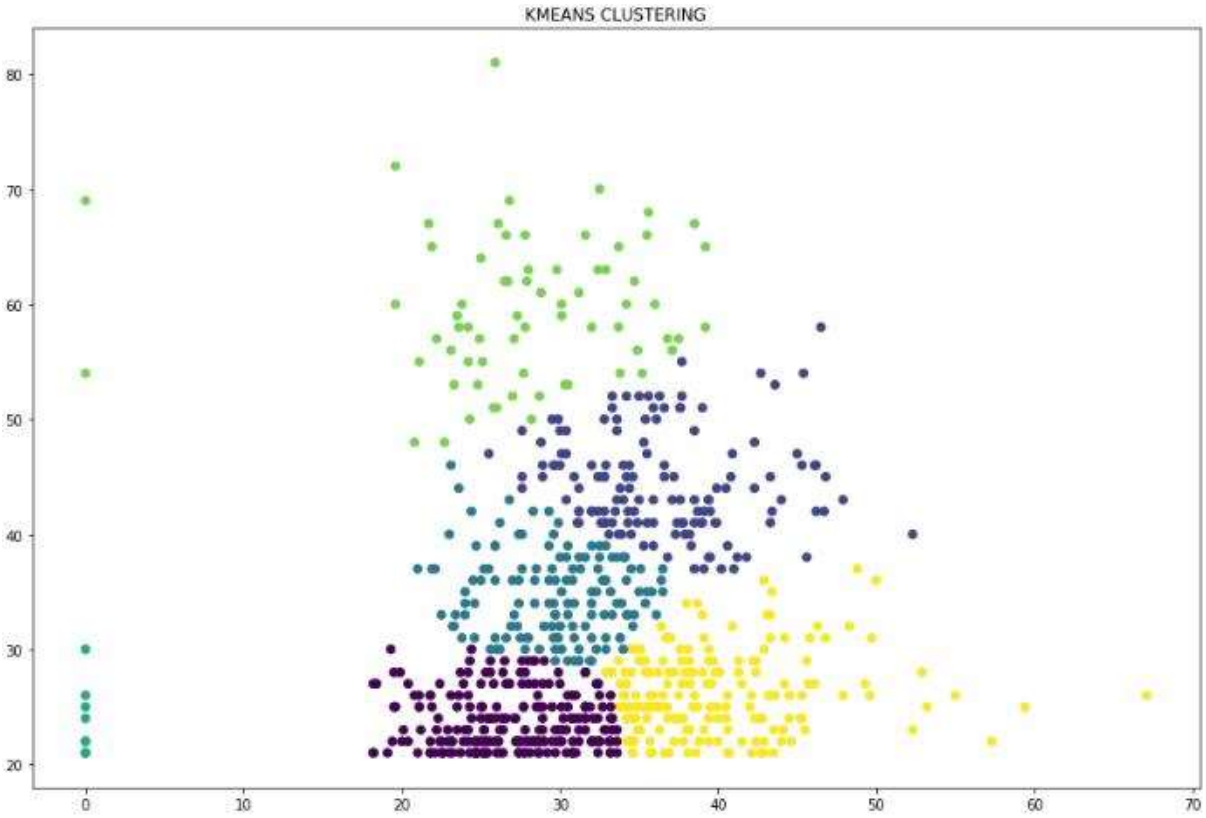
Con k = 4



Con k = 2


KMEANS CLUSTERING
80
70
60
50
40
30
20
0
10
20
30
40
50
60
70

Con k = 6


KMEANS CLUSTERING
80
70
60
50
40
30
20
0
10
20
30
40
50
60
70

Se eligió 4 clusters porque la agrupación y clasificación es mejor a simple vista por separación y colores, además con la ayuda del método Elbow esta elección se hace más concisa y clara.

Con **data.describe()** podemos la descripción estadística de los datos:

	preg	plas	pres	skin	insu	mass	pedi	age
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000

Como conclusión de la segunda parte, concluimos de manera práctica que usando los datos mass(masa), y age(edad) las personas de entre 25 a 40 años y con masa de entre 30 a 40 kg son más propensas a tener diabetes. Frente al uso del algoritmo K-means nos ayuda mucho a crear grupos o clústers cuando los datos son grandes como en este caso, así observar las relaciones que tienen los datos entre si.

Como conclusión final deducimos de ambas partes que las personas con las siguientes características son más propensas a padecer diabetes:

1. **Polyuria** (Producción de orina de > 3 L por día).
2. **Polydipsia** (Tener mucha sed y tomar una gran cantidad de líquido).
3. **Gender** (Género, siendo hombres con mayor número de registros, 63%).
4. **Alopecia** (Pérdida anormal del cabello).
5. **Age** (Edad, 25 a 40 años).
6. **Peso** (30 a 40 kg).