Minería de Datos **Ibañez Guzman Osvaldo**
Grupo: 1 **Semestre 2023-1**
Dr. Guillermo Gilberto Molero Castillo
Práctica #16: Clasificación - Árboles de decisión y Bosques aleatorios

Objetivo

Clasificar la mortalidad de adultos mayores contagiados con COVID-19 en la Ciudad de México.

Fuente de Información

https://www.gob.mx/salud/documentos/datos-abiertos-bases-historicas-direccion-general-de-epidemiologia

Fuente de Datos

- ID
- SEXO Identifica el sexo del paciente. 1-Mujer, 2-Hombre, 99-No Especificado
- TIPO_PACIENTE Identifica el tipo de atención que recibió el paciente. 1-Ambulatorio, 2-Hospitalizado, 99-No Especificado
- SITUACION Identifica la situación (vivo o muerto) del paciente. 1-Vivo 2-Muerto
- INTUBADO Identifica si el paciente requirió de intubación. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- NEUMONIA Identifica si el paciente se le diagnosticó con neumonía. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- EDAD Identifica la edad del paciente. Numérico
- DIABETES Identifica si el paciente tiene un diagnóstico de diabetes. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- EPOC Identifica si el paciente tiene un diagnóstico de Enfermedad Pulmonar Obstructiva Crónica (EPOC).1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- ASMA Identifica si el paciente tiene un diagnóstico de asma. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- INMUSUPPR Identifica si el paciente tiene un diagnóstico de inmunosupresión. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- HIPERTENSION Identifica si el paciente tiene un diagnóstico de hipertensión. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- OTRA_COM Identifica si el paciente tiene diagnóstico de otras enfermedades. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- CARDIOVASCULAR Identifica si el paciente tiene un diagnóstico de enfermedades cardiovasculares. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- OBESIDAD Identifica si el paciente tiene diagnóstico de obesidad. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- RENAL_CRONICA Identifica si el paciente tiene diagnóstico de insuficiencia renal crónica.1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- TABAQUISMO Identifica si el paciente tiene hábito de tabaquismo. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado
- OTRO_CASO Identifica si el paciente tuvo contacto con algún otro caso diagnosticado con SARS-CoV-2. 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado

- RESULTADO_ANTIGENO Identifica el resultado del análisis de la muestra de antígeno para SARS-CoV-2. 1-Positivo SARS-CoV-2, 2-Negativo SARS-CoV-2, 97- No Aplica (Caso sin muestra)
- CLASIFICACION_FINAL Identifica la clasificación del resultado de la prueba Covid-19: confirmado, inválido, no realizado, sospechoso y negativo. 1-Confirmado por Asociación Clínica Epidemiológica, 2-Confirmado por comité de Dictaminación, 3-Caso confirmado, 4-Inválido por laboratorio, 5-No realizado por laboratorio, 6-Caso sospechoso, 7-Negativo a SARS-CoV-2.
- UCI Identifica si el paciente requirió ingresar a una Unidad de Cuidados Intensivos (UCI). 1-Sí, 2-No, 97-No aplica, 98-Se ignora, 99-No Especificado

Desarrollo

La primer parte de la práctica consiste en el importación de bibliotecas, donde a cada una de las bibliotecas la renombraremos para funciones practicas ya que no será necesario escribir el nombre completo si no solo algunas letras, dentro de las bibliotecas a usar estarán “pandas” que ayuda al análisis y manipulación de datos, “numpy” para creación de matrices y vectores de diferentes dimensiones, “matplotlib.pyplot” que generara graficas con los datos recabados, “seaborn” para visualizar dichas gráficas y finalmente “matplotlib inline” para poder almacenar las gráficas dentro de nuestro entorno de desarrollo.

Los datos se encuentran en el archivo “CovidAdultosMatores.csv”, donde podremos encontrar 591352 registros o pacientes, con 21 columnas, dichas columnas ya se explicaron anteriormente.

```
Covid = pd.read_csv('CovidAdultosMayores.csv')
Covid
```

	ID	SEXO	TIPO_PACIENTE	SITUACION	INTUBADO	NEUMONIA	EDAD	DIABETES	EPOC	ASMA	...	HIPERTENSION	OTRA_COM	CARDIOVAS
0	21	2	1	Vivo	5	2	62	2	2	2	...	1	2	
1	23	1	1	Vivo	5	2	67	1	2	2	...	2	2	
2	31	1	1	Vivo	5	2	62	2	2	2	...	1	2	
3	39	1	1	Vivo	5	2	76	2	2	2	...	2	2	
4	81	1	1	Vivo	5	2	60	2	2	2	...	2	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
591347	15563006	1	1	Vivo	5	2	67	4	4	4	...	4	4	
591348	15563010	2	1	Vivo	5	2	68	4	4	4	...	4	4	
591349	15563012	2	1	Vivo	5	2	69	4	4	4	...	4	4	
591350	15563014	2	1	Vivo	5	2	71	4	4	4	...	4	4	
591351	15563019	1	1	Vivo	5	2	70	4	4	4	...	4	4	

591352 rows × 21 columns

El siguiente es la descripción de los datos, usando la función *info* donde podemos observar que contamos con todos los datos en todas las columnas, así como que todos los valores son de tipo numérico excepto la SITUACION, que es un objeto ya sea ‘Finado’ o ‘Vivo’.

```
Covid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 591352 entries, 0 to 591351
Data columns (total 21 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   ID                   591352 non-null  int64
 1   SEXO                 591352 non-null  int64
 2   TIPO_PACIENTE        591352 non-null  int64
 3   SITUACION            591352 non-null  object
 4   INTUBADO             591352 non-null  int64
 5   NEUMONIA             591352 non-null  int64
 6   EDAD                 591352 non-null  int64
 7   DIABETES             591352 non-null  int64
 8   EPOC                 591352 non-null  int64
 9   ASMA                 591352 non-null  int64
 10  INMUSUPR             591352 non-null  int64
 11  HIPERTENSION         591352 non-null  int64
 12  OTRA_COM             591352 non-null  int64
 13  CARDIOVASCULAR       591352 non-null  int64
 14  OBESIDAD             591352 non-null  int64
 15  RENAL_CRONICA        591352 non-null  int64
 16  TABAQUISMO           591352 non-null  int64
 17  OTRO_CASO            591352 non-null  int64
 18  RESULTADO_ANTIGENO   591352 non-null  int64
 19  CLASIFICACION_FINAL  591352 non-null  int64
 20  UCI                  591352 non-null  int64
dtypes: int64(20), object(1)
memory usage: 94.7+ MB
```

Además podemos observar que los datos aunque sean muchos, están desbalanceados, ya que contamos con muchos mas registros para pacientes con SITUACION 'Vivo', que 'Finado', además haciendo un análisis de nuestros datos podemos ver que la columna ID, no es relevante para nuestro análisis, así que no la tomaremos en cuenta, ya que podría resultar hasta contraproducente el usarla, haremos una descripción de los datos, para saber si se encuentran datos atípicos, en caso de existir podremos eliminar dichos registros si lo creemos conveniente.

```
print(Covid.groupby('SITUACION').size())
```

```
SITUACION
Finado     34296
Vivo      557056
dtype: int64
```

```
Covid = Covid.drop(columns='ID')
```

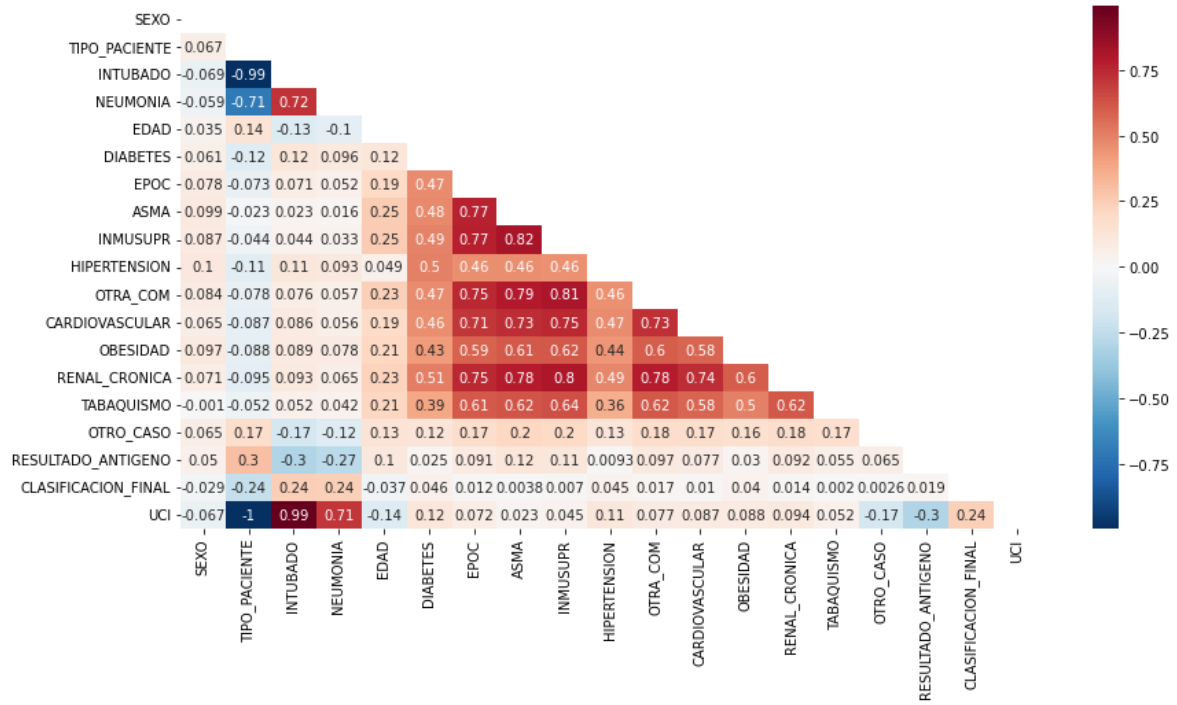
```
Covid.describe()
```

	SEXO	TIPO_PACIENTE	INTUBADO	NEUMONIA	EDAD	DIABETES	EPOC	ASMA	INMUSUPR	HIPERTENS
count	591352.000000	591352.000000	591352.000000	591352.000000	591352.000000	591352.000000	591352.000000	591352.000000	591352.000000	591352.000
mean	1.460714	1.117209	4.630892	1.917734	68.538033	1.812335	2.005494	2.014942	2.017462	1.718
std	0.498455	0.321670	1.020574	0.275912	8.773569	0.490601	0.282409	0.263831	0.258697	0.536
min	1.000000	1.000000	1.000000	1.000000	3.000000	1.000000	1.000000	1.000000	1.000000	1.000
25%	1.000000	1.000000	5.000000	2.000000	62.000000	2.000000	2.000000	2.000000	2.000000	1.000
50%	1.000000	1.000000	5.000000	2.000000	66.000000	2.000000	2.000000	2.000000	2.000000	2.000
75%	2.000000	1.000000	5.000000	2.000000	72.000000	2.000000	2.000000	2.000000	2.000000	2.000
max	2.000000	2.000000	5.000000	3.000000	122.000000	4.000000	4.000000	4.000000	4.000000	4.000

SELECCIÓN DE CARACTERISTICAS

El paso siguiente debe ser la selección de características, buscando que no existan correlaciones fuertes en los modelos, pero para esta practica utilizaremos todas las columnas ya que no sabemos cuales si son relevantes para los casos clínicos, posiblemente al analizar los daos con expertos en el tema podremos llegar a descartar ciertas variables, pero por el momento utilizaremos todas, sin importar la correlación que tenga.

```
plt.figure(figsize=(14,7))
MatrizInf = np.triu(Covid.corr())
sns.heatmap(Covid.corr(), cmap='RdBu_r', annot=True, mask=MatrizInf)
plt.show()
```



Entonces las variables predictoras serán todas nuestras columnas exceptuando SITUACION y ID, este ultimo como mencionamos no lo tomaremos en cuenta, y para la variable de clase usaremos Situación, que es la variable que queremos predecir, con los métodos de árbol de decisiones y bosques aleatorios de clasificación.

```
X = np.array(Covid[['SEXO',
                    'TIPO_PACIENTE',
                    'INTUBADO',
                    'NEUMONIA',
                    'EDAD',
                    'DIABETES',
                    'EPOC',
                    'ASMA',
                    'INMUSUPR',
                    'HIPERTENSION',
                    'OTRA_COM',
                    'CARDIOVASCULAR',
                    'OBESIDAD',
                    'RENAL_CRONICA',
                    'TABAQUISMO',
                    'OTRO_CASO',
                    'RESULTADO_ANTIGENO',
                    'CLASIFICACION_FINAL',
                    'UCI']])
pd.DataFrame(X)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	2	1	5	2	62	2	2	2	2	1	2	2	2	2	2	2	5	6	5
1	1	1	5	2	67	1	2	2	2	2	2	2	2	2	2	1	5	3	5
2	1	1	5	2	62	2	2	2	2	1	2	2	1	2	2	1	1	3	5
3	1	1	5	2	76	2	2	2	2	2	2	2	2	2	2	2	5	5	5
4	1	1	5	2	60	2	2	2	2	2	2	2	1	2	2	2	5	7	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
591347	1	1	5	2	67	4	4	4	4	4	4	4	4	4	4	1	2	7	5
591348	2	1	5	2	68	4	4	4	4	4	4	4	4	4	4	1	2	7	5
591349	2	1	5	2	69	4	4	4	4	4	4	4	4	4	4	1	2	7	5
591350	2	1	5	2	71	4	4	4	4	4	4	4	4	4	4	1	2	7	5
591351	1	1	5	2	70	4	4	4	4	4	4	4	4	4	4	1	2	7	5

591352 rows × 19 columns

```
Y = np.array(Covid[['SITUACION']])
pd.DataFrame(Y)
```

	0
0	Vivo
1	Vivo
2	Vivo
3	Vivo
4	Vivo
...	...
591347	Vivo
591348	Vivo
591349	Vivo
591350	Vivo
591351	Vivo

591352 rows × 1 columns

Es necesario separar tanto nuestra variables predictoras, como nuestra variable de clase, una vez separadas con las características que queremos, observaremos la longitud de estas y tenemos 473081 registros para el entrenamiento y 118271 para la validación de nuestros datos.

```
X_train, X_validation, Y_train, Y_validation = model_selection.train_test_split(X, Y,
                                                                                test_size = 0.2,
                                                                                random_state = 0,
                                                                                shuffle = True)
```

```
print(len(X_train))
print(len(X_validation))
```

```
473081
118271
```

MODELO ARBOLES DE DECISION

Para crear el modelo del árbol de decisión, utilizaremos la biblioteca *DecisionTreeClassifier* ya que recordaremos que haremos una clasificación y no un pronóstico, ya que no buscamos un valor si no únicamente saber si la SITUACION del paciente. El árbol que usaremos contara con un máximo de 14 niveles, así como un mínimo de elementos por hoja y un mínimo de hijos.

```
from sklearn.tree import DecisionTreeClassifier
```

```
ClasificacionAD = DecisionTreeClassifier(max_depth=14,
                                         min_samples_split=4,
                                         min_samples_leaf=2,
                                         random_state=0)
ClasificacionAD.fit(X_train, Y_train)
```

```
DecisionTreeClassifier(max_depth=14, min_samples_leaf=2, min_samples_split=4,
                       random_state=0)
```

Lo siguiente es obtener los valores que nuestro modelo predijo, esto para obtener la matriz de clasificación comparando los valores reales con los obtenidos con el modelo, en dicha matriz observamos que se clasificó correctamente a 4100 Finados y 109197 Vivos, mientras que clasifico de forma incorrecta a 4974 registros.

```
Y_ClasificacionAD = ClasificacionAD.predict(X_validation)
print(Y_ClasificacionAD)
```

```
['Vivo' 'Vivo' 'Vivo' ... 'Vivo' 'Vivo' 'Vivo']
```

```
#Matriz de clasificación
ModeloClasificacion1 = ClasificacionAD.predict(X_validation)
Matriz_Clasificacion1 = pd.crosstab(Y_validation.ravel(),
                                    ModeloClasificacion1,
                                    rownames=['Reales'],
                                    colnames=['Clasificación'])
Matriz_Clasificacion1
```

Clasificación	Finado	Vivo
Reales		
Finado	4100	2730
Vivo	2244	109197

Así también podemos obtener el reporte de clasificación, donde entre otras cosas podemos observar la importancia que tienen las diferentes variables, así como la exactitud de nuestro modelo, y la precisión para cada una de las clasificaciones que realiza.

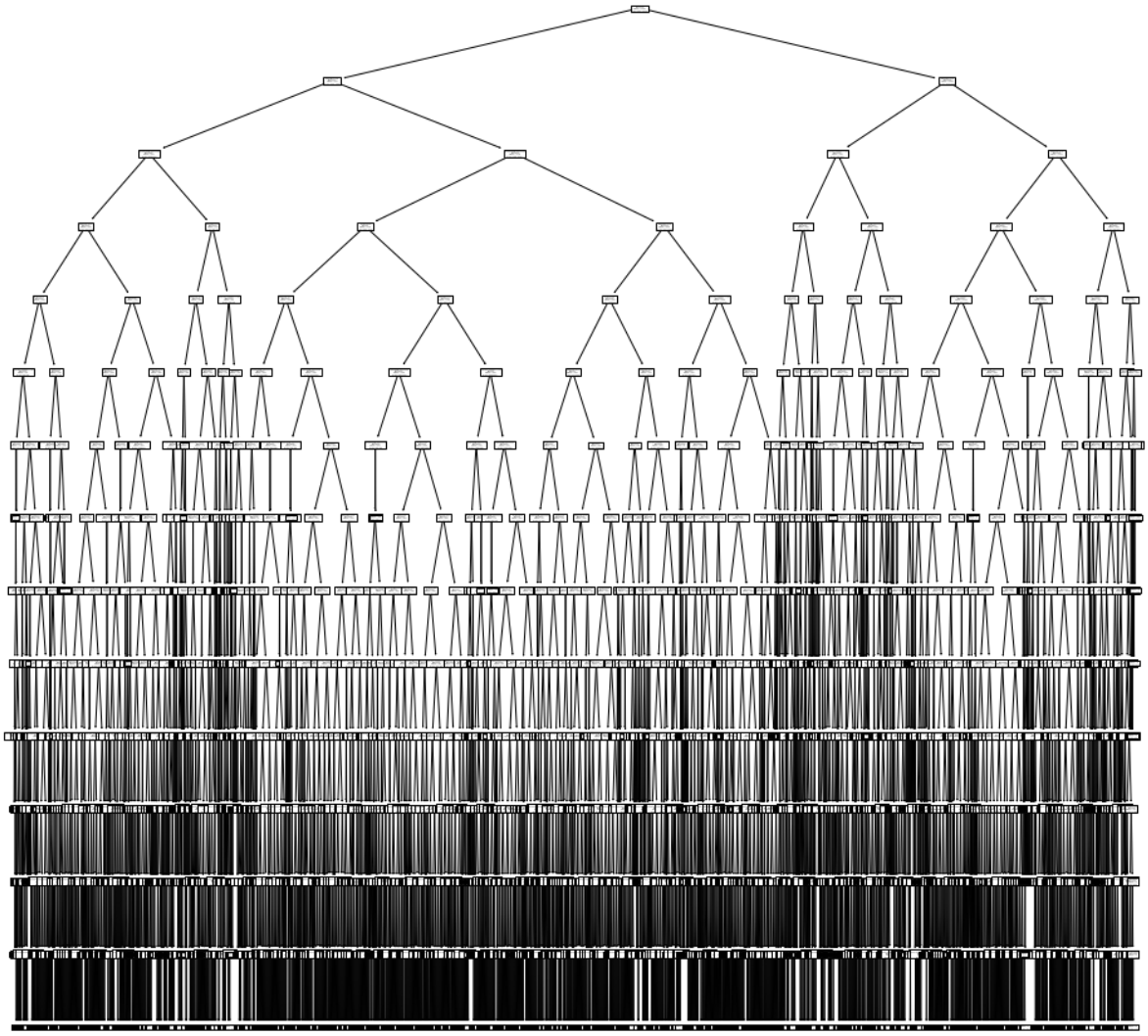
```
print('Criterio: \n', ClasificacionAD.criterion)
print('Importancia variables: \n', ClasificacionAD.feature_importances_)
print("Exactitud:", accuracy_score(Y_validation, Y_ClasificacionAD))
print(classification_report(Y_validation, Y_ClasificacionAD))
```

```
Criterio:
 gini
Importancia variables:
 [0.00762704 0.         0.7637445  0.02132563 0.04133053 0.00624171
 0.00297293 0.00141851 0.00194786 0.00633222 0.00359294 0.00386185
 0.00477569 0.00328715 0.00409712 0.01154335 0.01229011 0.09939872
 0.00421215]
Exactitud: 0.9579440437638982
              precision    recall  f1-score   support

      Finado       0.65      0.60      0.62      6830
        Vivo       0.98      0.98      0.98    111441

    accuracy                           0.96    118271
   macro avg       0.81      0.79      0.80    118271
weighted avg       0.96      0.96      0.96    118271
```

La forma más cómoda para observar la importancia de nuestras variables es colocarla en una tabla, en dicha tabla observamos que la variable con mas peso es INTUBADO, que influye casi un 76% en la clasificación asignada, y así de forma descendente, hasta la que tiene menos importancia en este caso es TIPO_PACIENTE.

	Variable	Importancia
2	INTUBADO	0.763744
17	CLASIFICACION_FINAL	0.099399
4	EDAD	0.041331
3	NEUMONIA	0.021326
16	RESULTADO_ANTIGENO	0.012290
15	OTRO_CASO	0.011543
0	SEXO	0.007627
9	HIPERTENSION	0.006332
5	DIABETES	0.006242
12	OBESIDAD	0.004776
18	UCI	0.004212
14	TABAQUISMO	0.004097
11	CARDIOVASCULAR	0.003862
10	OTRA_COM	0.003593
13	RENAL_CRONICA	0.003287
6	EPOC	0.002973
8	INMUSUPR	0.001948
7	ASMA	0.001419
1	TIPO_PACIENTE	0.000000

Además mostráremos el árbol creado, que por el tamaño es un poco complicado observar los valores que tiene pero nos ayuda a poder observar la estructura que tiene nuestro árbol.



BOSQUES ALEATORIOS

Para hacer le modelo de bosques aleatorios, el proceso es bastante parecido al de arboles aleatorios, ya que usaremos las mismas variables predictoras y de clase, únicamente usaremos la función *RandomForestClassifier*, donde las características serán similares que en el caso de árboles, para que al momento de hacer una comparación las condiciones iniciales sean parecidas y no influya en el resultado.

```
from sklearn.ensemble import RandomForestClassifier
```

```
ClasificacionBA = RandomForestClassifier(n_estimators=100,
                                         max_depth=14,
                                         min_samples_split=4,
                                         min_samples_leaf=2,
                                         random_state=0)
ClasificacionBA.fit(X_train, Y_train)
```

Hacemos el arreglo de predicciones para el modelo de bosques aleatorios, y obtenemos su matriz de clasificación, donde tenemos un total de 4650 registros clasificados de forma incorrecta, mientras que los restantes de forma correcta.

```
Y_ClasificacionBA = ClasificacionBA.predict(X_validation)
print(Y_ClasificacionBA)
```

```
['Vivo' 'Vivo' 'Vivo' ... 'Vivo' 'Vivo' 'Vivo']
```

```
ModeloClasificacion2 = ClasificacionBA.predict(X_validation)
Matriz_Clasificacion2 = pd.crosstab(Y_validation.ravel(),
                                    ModeloClasificacion2,
                                    rownames=['Reales'],
                                    colnames=['Clasificación'])
Matriz_Clasificacion2
```

Clasificación	Finado	Vivo
Reales		
Finado	3927	2903
Vivo	1747	109694

En cuanto al reporte de clasificación donde de igual forma observamos la importancia de las variables, así como la exactitud del modelo y la precisión para cada una de las clasificaciones que hace.

```
print('Criterio: \n', ClasificacionBA.criterion)
print('Importancia variables: \n', ClasificacionBA.feature_importances_)
print("Exactitud:", accuracy_score(Y_validation, Y_ClasificacionBA))
print(classification_report(Y_validation, Y_ClasificacionBA))
```

```
Criterio:
 gini
Importancia variables:
 [0.00201745 0.25911669 0.32715073 0.14159703 0.00788453 0.00109819
 0.00061653 0.00041403 0.00036089 0.00097653 0.00107159 0.00091694
 0.00064092 0.00240605 0.00042331 0.00900338 0.0351927  0.06622155
 0.14289099]
Exactitud: 0.9606835149783125
              precision    recall  f1-score   support

      Finado       0.69      0.57      0.63      6830
        Vivo       0.97      0.98      0.98    111441

    accuracy                           0.96    118271
   macro avg       0.83      0.78      0.80    118271
weighted avg       0.96      0.96      0.96    118271
```

Al observar la importancia de las variables, observamos que INTUBADO sigue siendo la que tiene mas peso, pero no tanto como en los bosques aleatorios, además de que en este caso TIPO_PACIENTE es una de las más importantes, a comparación del otro caso que no tenia ninguna importancia.

	Variable	Importancia
2	INTUBADO	0.270618
1	TIPO_PACIENTE	0.212354
18	UCI	0.209908
3	NEUMONIA	0.114815
17	CLASIFICACION_FINAL	0.086861
4	EDAD	0.027222
16	RESULTADO_ANTIGENO	0.026243
15	OTRO_CASO	0.013814
0	SEXO	0.005632
5	DIABETES	0.005057
9	HIPERTENSION	0.004166
13	RENAL_CRONICA	0.004130
12	OBESIDAD	0.003649
11	CARDIOVASCULAR	0.003479
14	TABAQUISMO	0.003255
10	OTRA_COM	0.002904
6	EPOC	0.002783
8	INMUSUPR	0.001777
7	ASMA	0.001331

VALIDACIÓN

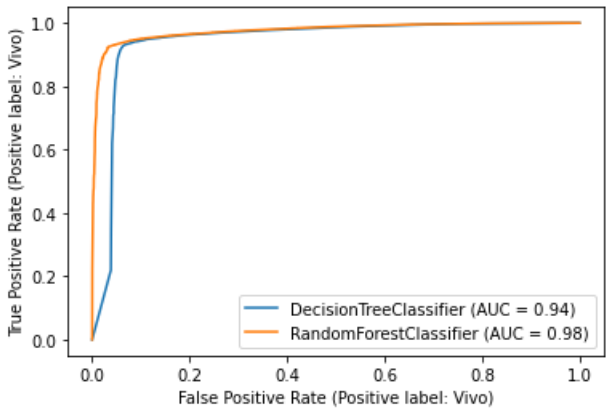
Verificaremos le exactitud que se lesa da a cada uno de nuestros modelos, donde podemos ver que prácticamente tienen el mismo puntaje, pero el bosque aleatorio tiene un poco mas de exactitud ya que el rango va de 0 a 1, donde 1 significa el 100% de exactitud de nuestro modelo.

```
print("Árbol de decisión:", accuracy_score(Y_validation, Y_ClasificacionAD))
print("Bosque aleatorio:", accuracy_score(Y_validation, Y_ClasificacionBA))
```

```
Árbol de decisión: 0.9579440437638982
Bosque aleatorio: 0.9613937482561237
```

Otra forma de observarlo es con las siguientes curvas, donde en otras palabras entre mas recto sea el ángulo (90°) qué forma se vuelve más exacto, así es como podemos observar que la recta naranja (Bosque Aleatorio) es mas exacto que la gráfica en azul (Árbol de Decisión)

```
from sklearn.metrics import RocCurveDisplay
from sklearn import metrics
fig, ax = plt.subplots()
RocCurveDisplay.from_estimator(ClasificacionAD,
                               X_validation,
                               Y_validation,
                               ax = ax)
metrics.RocCurveDisplay.from_estimator(ClasificacionBA,
                                       X_validation,
                                       Y_validation,
                                       ax = ax)
plt.show()
```



PRUEBAS

Para probar nuestros modelos, usaremos los datos de u paciente de nuestra tabla, por lo que le asignaremos os valores con los que ya cuenta, esperando que el resultado en este caso sea Finado, ya que es lo que tiene en el registro. Para este paciente podemos ver que ambos modelos dieron la clasificación correcta, por lo que podemos ver que los modelos son correctos, pero recordaremos que no son 100% exactos.

```
# Paciente 2155 - FINADO
PacienteAD = pd.DataFrame({'SEXO': [2],
                           'TIPO_PACIENTE': [2],
                           'INTUBADO': [2],
                           'NEUMONIA': [2],
                           'EDAD': [84],
                           'DIABETES': [2],
                           'EPOC': [2],
                           'ASMA': [2],
                           'INMUSUPR': [2],
                           'HIPERTENSION': [1],
                           'OTRA_COM': [2],
                           'CARDIOVASCULAR': [2],
                           'OBESIDAD': [2],
                           'RENAL_CRONICA': [2],
                           'TABAQUISMO': [2],
                           'OTRO_CASO': [2],
                           'RESULTADO_ANTIGENO': [1],
                           'CLASIFICACION_FINAL': [3],
                           'UCI': [2]})
ClasificacionAD.predict(PacienteAD)
```

```
D:\Users\osva_\anaconda3\lib\site-packages\sklearn\
s fitted without feature names
  warnings.warn(
```

```
array(['Finado'], dtype=object)
```

```
# Paciente 2155 - FINADO
PacienteBA = pd.DataFrame({'SEXO': [2],
                           'TIPO_PACIENTE': [2],
                           'INTUBADO': [2],
                           'NEUMONIA': [2],
                           'EDAD': [84],
                           'DIABETES': [2],
                           'EPOC': [2],
                           'ASMA': [2],
                           'INMUSUPR': [2],
                           'HIPERTENSION': [1],
                           'OTRA_COM': [2],
                           'CARDIOVASCULAR': [2],
                           'OBESIDAD': [2],
                           'RENAL_CRONICA': [2],
                           'TABAQUISMO': [2],
                           'OTRO_CASO': [2],
                           'RESULTADO_ANTIGENO': [1],
                           'CLASIFICACION_FINAL': [3],
                           'UCI': [2]})
ClasificacionBA.predict(PacienteBA)
```

```
D:\Users\osva_\anaconda3\lib\site-packages\sklearn\
s fitted without feature names
  warnings.warn(
```

```
array(['Finado'], dtype=object)
```

Conclusión

Es interesante ver como el procesamiento de los datos es bastante similar y únicamente cambiamos el modelo a utilizar y eso puede mejorar la exactitud de nuestro modelo, además de que posiblemente si se reduce la cantidad de variables o se cambian las condiciones iniciales de los modelos podamos tener diferentes resultados, ya sean mas o menos precisos al mostrado en este documento.

Liga GitHub Practica 16

Cuaderno de Python de la practica 16

https://github.com/OsvaldoIG/MineriaDatos/tree/main/P16