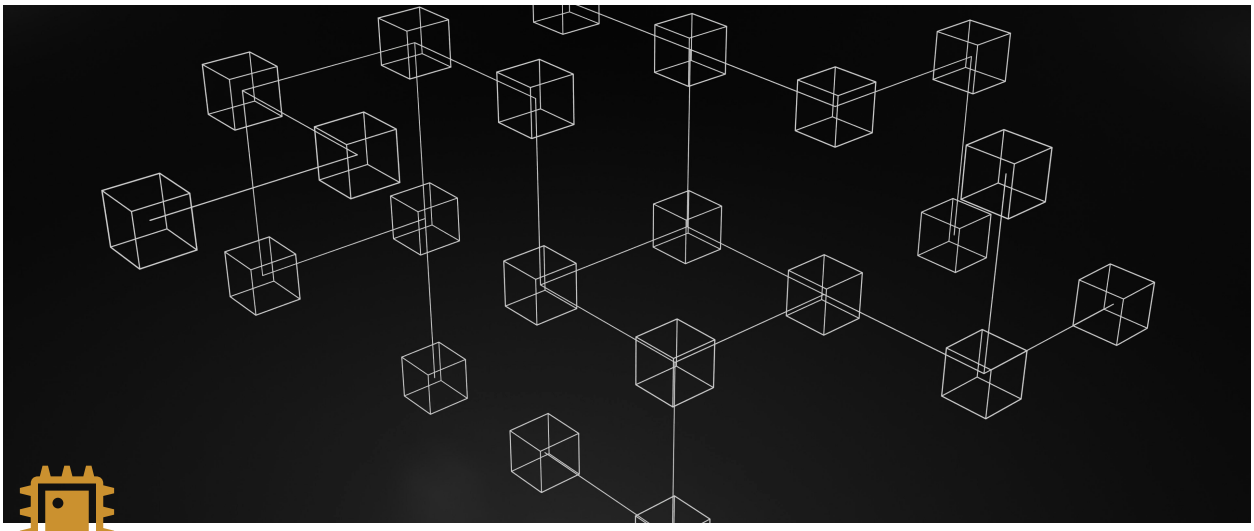
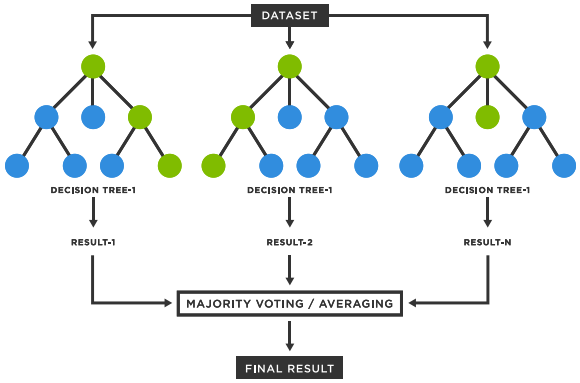


Modelado y Evaluación

Técnica del modelado

Para nuestro proyecto, era muy importante que de todos los contratos que ha adjudicado el estado se clasificara que contratos fueron cerrados y de esta forma poder realizar un modelo que pudiera predecir ésta clasificación `cerrado` según los parámetros definidos para su entrenamiento.



En Machine Learning se cuentan con varios algoritmos de clasificación de aprendizaje supervisado, tales como K-Nearest Neighbors, Regresión Logística, Naive Bayes, Árboles de desiciones, Random Forest y Redes Neuronales. Para nuestro proyecto decidimos utilizar Random Forest como algoritmo de clasificación

```
from sklearn.model_selectio
n import train_test_split
```

debido a que nos ofrece la clasificación árboles de decisión que participan en si para la clasificación de un determinado elemento.

```
train_x, rest_x, train_y, r
est_y = train_test_split(co
ntratos_data, estado_contra
to, train_size=training_cou
nt)
test_x, validate_x, test_y,
validate_y = train_test_spl
it(rest_x, rest_y, train_si
ze=test_count)
```

Conjuntos de Entrenamiento, Pruebas y Validación

Una vez tratado el archivo original de los contratos del SECOP II, obtuvimos un DataFrame de 549.407 registros con 34 parámetros; de este DataFrame decidimos tomar el 60% de los registros para entrenamiento, el 20% para pruebas y el otro 20% para validación.

```
# Calculate test and valida
tion set size:
original_count = len(df_fin
al)
training_size = 0.60 # 60%
of records
test_size = (1 - training_s
ize) / 2

training_count = int(origin
al_count * training_size)
test_count = int(original_c
ount * test_size)
validation_count = original
```

Conjunto de datos y sus registros

Conjunto de entrenamiento	329.644
Conjunto de pruebas	109.881
Conjunto de Validación	109.882

```
_count - training_count - t
est_count
```

Evaluación del Modelo

Para la evaluación del modelo utilizamos dos métricas de clasificación para medir que tan bien hace las predicciones con los parámetros definidos para el nuestros algoritmo de clasificación Random Forest.

Accurancy

Usamos esta métrica para determinar la exactitud de las predicciones correctas sobre el total de predicciones realizadas

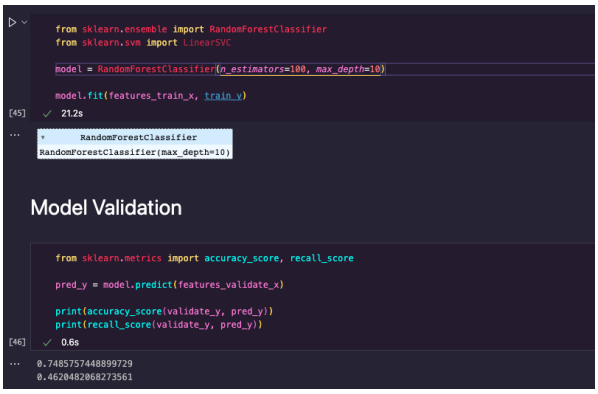
Recall

Utilizamos esta métrica para medir la capacidad del modelo de identificar todos los casos positivos (verdaderos positivos)

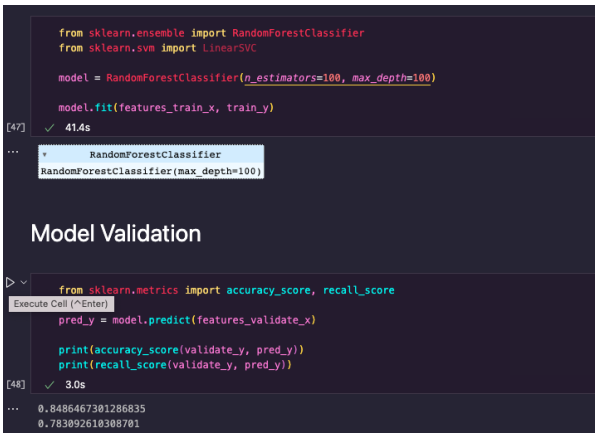
Análisis de entrenamiento del modelo

Inicialmente validamos nuestro modelo usando 100 estimadores (# de árboles) y tomando los demás parámetros por default del algoritmo

`RandomForestClassifier`; esto nos dio un accurancy de **0.8489** y un recall de **0.7846** que son buenos indicadores para el modelo ya que están cercanos a 1.



Luego entrenamos nuestro modelo usando 200 estimadores y tomando los demás parámetros por default del algoritmo; esto nos dio un accurancy de **0.8490** y un recall de **0.7847** que no son un incremento significativo de nuestras métricas iniciales y con un tiempo de entrenamiento mayor al inicial.



Ya analizado que 100 estimadores era el parámetro ideal, procedimos a entrenar nuestro modelo usando 100 estimadores, con una profundidad de 10 (# nodos/hojas de profundidad del árbol) y tomando los demás parámetros por default del algoritmo; esto nos dio un accurancy de **0.7485** y un recall de **0.4620** que nos indica un empeoramiento en las métricas.

Con el experimento anterior, decidimos aumentar la profundidad a 100 y esto dio un accurancy de **0.8486** y un recall de **0.7830** que mejoran las métricas y se acercan a las iniciales.

Parámetros finales

Una vez realizado todos estos experimentos donde realizamos un juego de ejecuciones cambiando los parámetros de estimadores y profundidad, llegamos a la conclusión que entrenar el modelo con 100 estimadores y los démas parámetros en default era suficiente para nuestro modelo de clasificación.