

# **Clasificación de aguas superficiales**

**Ciencia y Analítica de Datos**

Dante Rodrigo Serna Camarillo | **A01182676**

# Análisis

- Características del dataset:

- 4,141 registros para 54 variables y una salida (Semáforo)
- 648 registros sin variable de salida ( 15.65 % )

```
# de Registros sin valor en variable de salida: : 648 ( 15.65 %)
# de Registros por categorización en semáforo:

Verde      1267
Amarillo   1135
Rojo       1091
Name: SEMAFORO, dtype: int64
```

- Las siguientes columnas tiene una relación directa, pues una columna guarda la magnitud de la muestra para determinado indicador mientras que la segunda denota una clasificación categórica para dicho valor.

Variable con medición	Variable categórica
DBO_mg/L	CALIDAD_DBO
DQO_mg/L	CALIDAD_DQO
SST_mg/L	CALIDAD_SST
COLI_FEC_NMP_100mL	CALIDAD_COLI_FEC
E_COLI_NMP_100mL	CALIDAD_E_COLI
ENTEROC_NMP_100mL	CALIDAD_ENTEROC
OD_PORC	CALIDAD_OD_PORC
OD_PORC_SUP	CALIDAD_OD_PORC_SUP
OD_PORC_MED	CALIDAD_OD_PORC_MED
OD_PORC_FON	CALIDAD_OD_PORC_FON
TOX_D_48_UT	CALIDAD_TOX_D_48
TOX_V_15_UT	CALIDAD_TOX_V_15
TOX_D_48_SUP_UT	CALIDAD_TOX_D_48_SUP
TOX_D_48_FON_UT	CALIDAD_TOX_D_48_FON
TOX_FIS_SUP_15_UT	CALIDAD_TOX_FIS_SUP_15
TOX_FIS_FON_15_UT	CALIDAD_TOX_FIS_FON_15

- Columnas completamente nulas:

- *TOX\_D\_48\_FON\_UT*
- *CALIDAD\_TOX\_D\_48\_FON*
- *TOX\_FIS\_FON\_15\_UT*
- *CALIDAD\_TOX\_FIS\_FON\_15*

- Columna de [**PERIODO**] cuenta con solo un valor: [**2020**]
- Columna de [**CLAVE**] es un identificador único para el cuerpo de agua.
- Columnas de [**ESTADO, MUNICIPIO**] pueden ser derivadas a través de latitud y longitud. [**LONGITUD, LATITUD**]
- En las columnas de variables numéricos/continuos, algunos valores están guardados con expresiones -mayor que, menor que- [<, >]

- Distribución de clases:  
(Registros clasificados)

SEMAFORO DISTRIBUTION		
0	Verde	0.362725
1	Amarillo	0.324936
2	Rojo	0.312339

# Limpieza

- Eliminar registros sin variable de salida/clasificación.

```
#Creamos nuevo dataset y eliminamos registros sin datos de salida.  
dfForCleansingSuperficiales = dfUnprocessedSuperficiales.copy()  
dfForCleansingSuperficiales = dfForCleansingSuperficiales[dfForCleansingSuperficiales['SEMAFORO'].notna()]  
dfForCleansingSuperficiales.shape
```

- Eliminar columnas completamente vacías

```
dfForCleansingSuperficiales.drop('TOX_D_48_FON_UT', inplace=True, axis=1)  
dfForCleansingSuperficiales.drop('CALIDAD_TOX_D_48_FON', inplace=True, axis=1)  
dfForCleansingSuperficiales.drop('TOX_FIS_FON_15_UT', inplace=True, axis=1)  
dfForCleansingSuperficiales.drop('CALIDAD_TOX_FIS_FON_15', inplace=True, axis=1)
```

- Remover símbolos [<, >] de los valores para hacer casting a *float* datatype (De acuerdo a la especificación de las columnas, remover estos símbolos aún sería consistente con las categorías.)

```
dfForCleansingSuperficiales = dfForCleansingSuperficiales.astype({'DBO_mg/L': float, 'DQO_mg/L': float, 'SST_mg/L': float, 'COLI_FEC_NMP_100mL': float,  
'E_COLI_NMP_100mL': float, 'ENTEROC_NMP_100mL': float, 'OD_PORC': float, 'OD_PORC_SUP': float, 'OD_PORC_MED': float, 'OD_PORC_FON': float,  
'TOX_D_48_UT': float, 'TOX_V_15_UT': float, 'TOX_D_48_SUP_UT': float, 'TOX_FIS_SUP_15_UT': float })
```

- Eliminar la columna de [PERIODO].
- Eliminar columnas que están relacionadas con la ubicación y pueden ser derivadas de [LONGITUD, LATITUD]
- Generamos dos data sets para comparar rendimiento de los modelos (Un set con valores categóricas y otro con valores continuos/numéricos)

```
dfForCleansingSuperficiales.drop('PERIODO', inplace=True, axis=1)
```

# Pipeline

- Imputación de valores faltantes. Definimos una nueva categoría para variables categóricas y usamos el valor -1 para las variables continuas.

```
[ ] #vamos a imputar los valores faltantes en las variables categoricos
columnasCategoricas = ['CALIDAD_DBO','CALIDAD_DQO','CALIDAD_SST','CALIDAD_COLI_FEC','CALIDAD_E_COLI','CALIDAD_ENTEROC','CALIDAD_OD_PORC',
'CALIDAD_OD_PORC_SUP','CALIDAD_OD_PORC_MED','CALIDAD_OD_PORC_FON','CALIDAD_TOX_D_48','CALIDAD_TOX_V_15','CALIDAD_TOX_D_48_SUP',
'CALIDAD_TOX_FIS_SUP_15','CONTAMINANTES']
imputerCategoricas = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='Sin medida')

columnasContinuas = ['DBO_mg/L','DQO_mg/L','SST_mg/L','COLI_FEC_NMP_100mL','E_COLI_NMP_100mL','ENTEROC_NMP_100mL','OD_PORC',
'OD_PORC_SUP','OD_PORC_MED','OD_PORC_FON','TOX_D_48_UT','TOX_V_15_UT','TOX_D_48_SUP_UT','TOX_FIS_SUP_15_UT']

imputerContinuas = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value=-1)

[ ] #definicion del column transformer
transformCategoricas = ColumnTransformer([('imputerCat', imputerCategoricas, columnasCategoricas)])
transformContinuas = ColumnTransformer([('imputerCon', imputerContinuas, columnasContinuas)])
```

- Label encoding para nuestra variable de salida.

```
[ ] #instanciar label encoder y entrenar con nuestros valores de semaforo
labelEncoder = preprocessing.LabelEncoder()
labelEncoder.fit(valoresSemaforo)

[ ] #aplicamos la transformacion a nuestra variable de salida en los dataframes.
#transform hace encoding, inverse_transform hace decoding
dfSuperficialesCategoricas.SEMAFORO = labelEncoder.transform(dfSuperficialesCategoricas.SEMAFORO)
dfSuperficialesContinuas.SEMAFORO = labelEncoder.transform(dfSuperficialesContinuas.SEMAFORO)
```

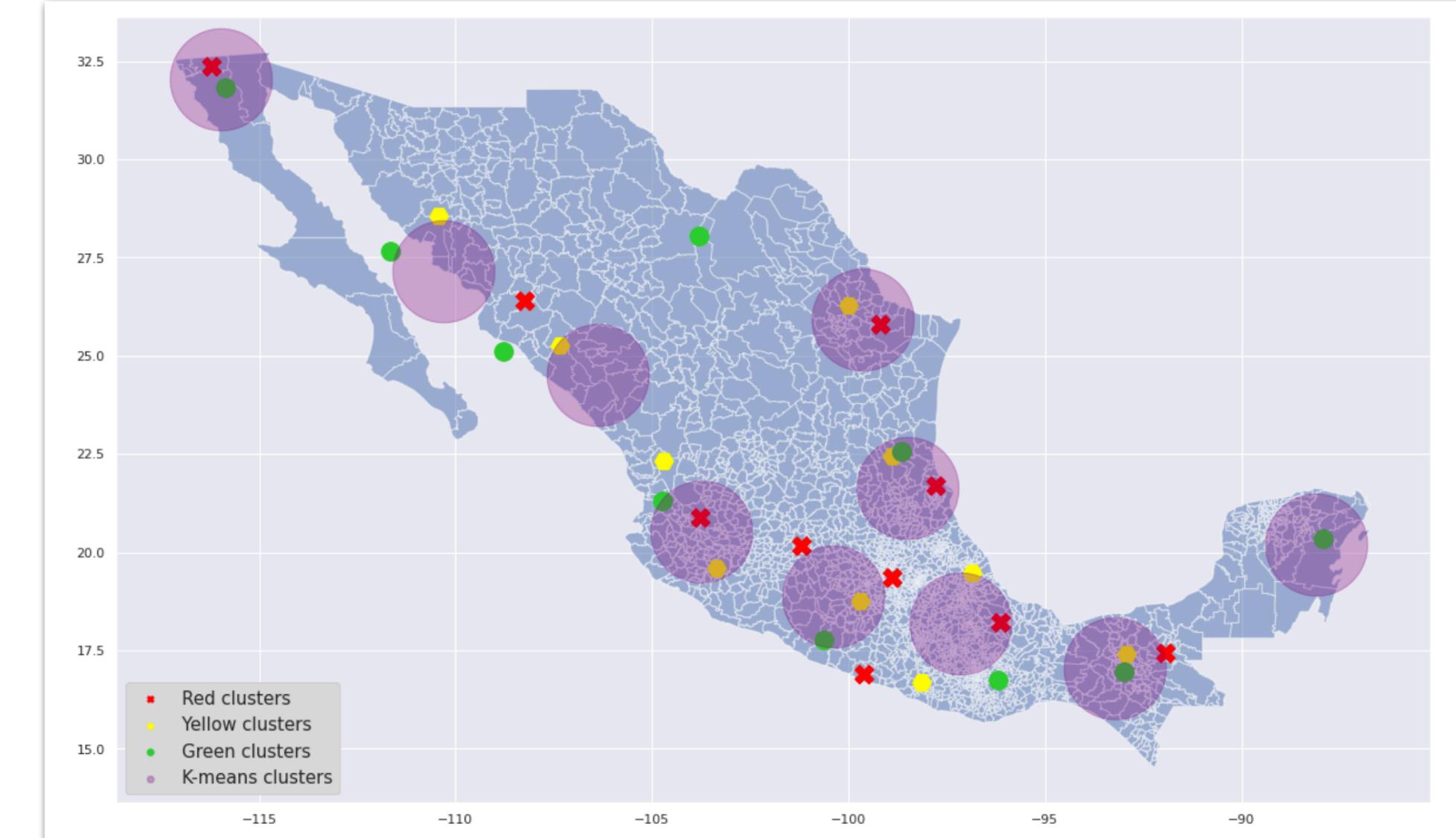
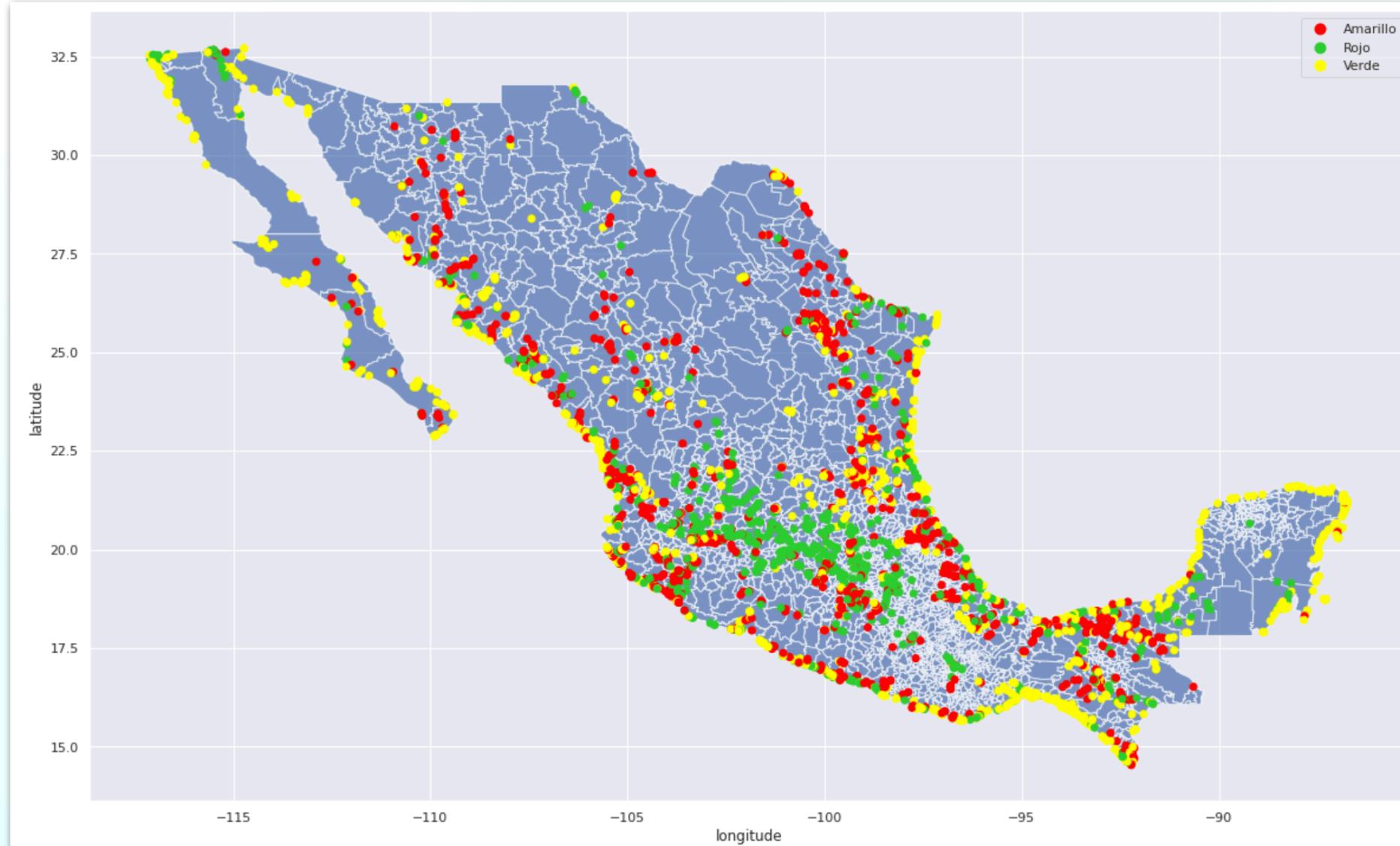
- Ordinal encoding (nos ayuda a contener la dimensión del dataset vs OneHot Encoding)

```
▶ #label encoding de variables categoricas
labelEncodingPipeline = Pipeline(steps = [('labelEncoding', OrdinalEncoder( dtype=np.str_, handle_unknown='use_encoded_value', unknown_value=-1 ))])
dfCatCategoricas = ['CALIDAD_DBO', 'CALIDAD_DQO', 'CALIDAD_SST', 'CALIDAD_COLI_FEC', 'CALIDAD_E_COLI', 'CALIDAD_ENTEROC', 'CALIDAD_OD_PORC', 'CALIDAD_OD_PORC_SUP', 'CALIDAD_OD_PORC_MED',
'CALIDAD_OD_PORC_FON', 'CALIDAD_TOX_D_48', 'CALIDAD_TOX_V_15', 'CALIDAD_TOX_D_48_SUP', 'CALIDAD_TOX_FIS_SUP_15', 'CONTAMINANTES', 'CUMPLE_CON_DBO', 'CUMPLE_CON_DQO', 'CUMPLE_CON_SST',
'CUMPLE_CON_CF', 'CUMPLE_CON_E_COLI', 'CUMPLE_CON_ENTEROC', 'CUMPLE_CON_OD', 'CUMPLE_CON_TOX', 'GRUPO']
dfConCategoricas = ['CONTAMINANTES', 'CUMPLE_CON_DBO', 'CUMPLE_CON_DQO', 'CUMPLE_CON_SST', 'CUMPLE_CON_CF',
'CUMPLE_CON_E_COLI', 'CUMPLE_CON_ENTEROC', 'CUMPLE_CON_OD', 'CUMPLE_CON_TOX', 'GRUPO']

# Conjuntamos las transformaciones numéricas y categóricas que se estarán aplicando a los datos de entrada:
featuresCatTransformer = ColumnTransformer(transformers = [('le', labelEncodingPipeline, dfCatCategoricas)],remainder='passthrough')
featuresConTransformer = ColumnTransformer(transformers = [('le', labelEncodingPipeline, dfConCategoricas)],remainder='passthrough')
```

# K-means & Clustering

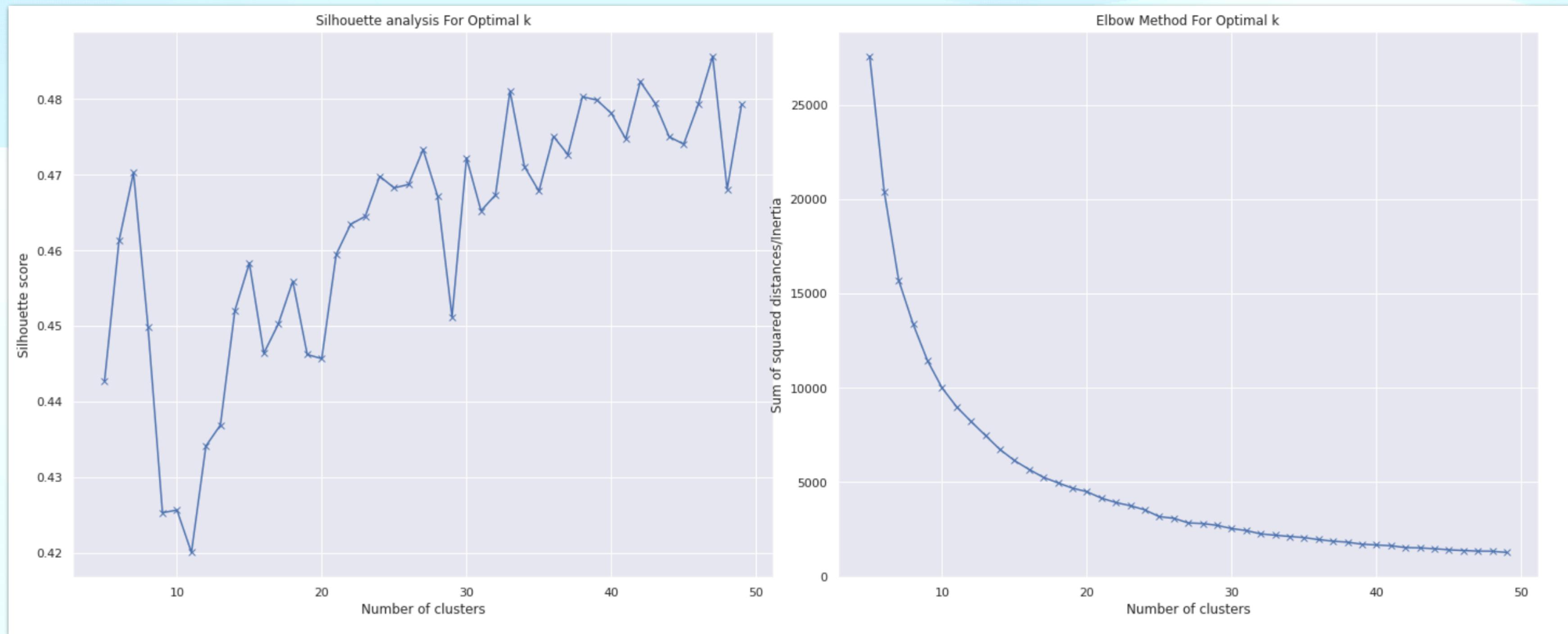
- Utilizamos K-means para identificar si se puede denotar alguna relación entre la clasificación del semáforo y los clusters generados.
- Graficamos los cuerpos de agua con un código de color de acuerdo a su semáforo como referencia, podemos notar que hay zonas con una presencia mayoritaria de algún semáforo y otros donde las segmentación no es tan evidente.
- En un primer baseline consideramos usar 10 clusters, podemos notar que la mayoría de los clusters capturarán puntos pertenecientes a dos o tres clases (utilizamos clusters de cada clase en específico para comparar la aproximación).



# K-means & Clustering

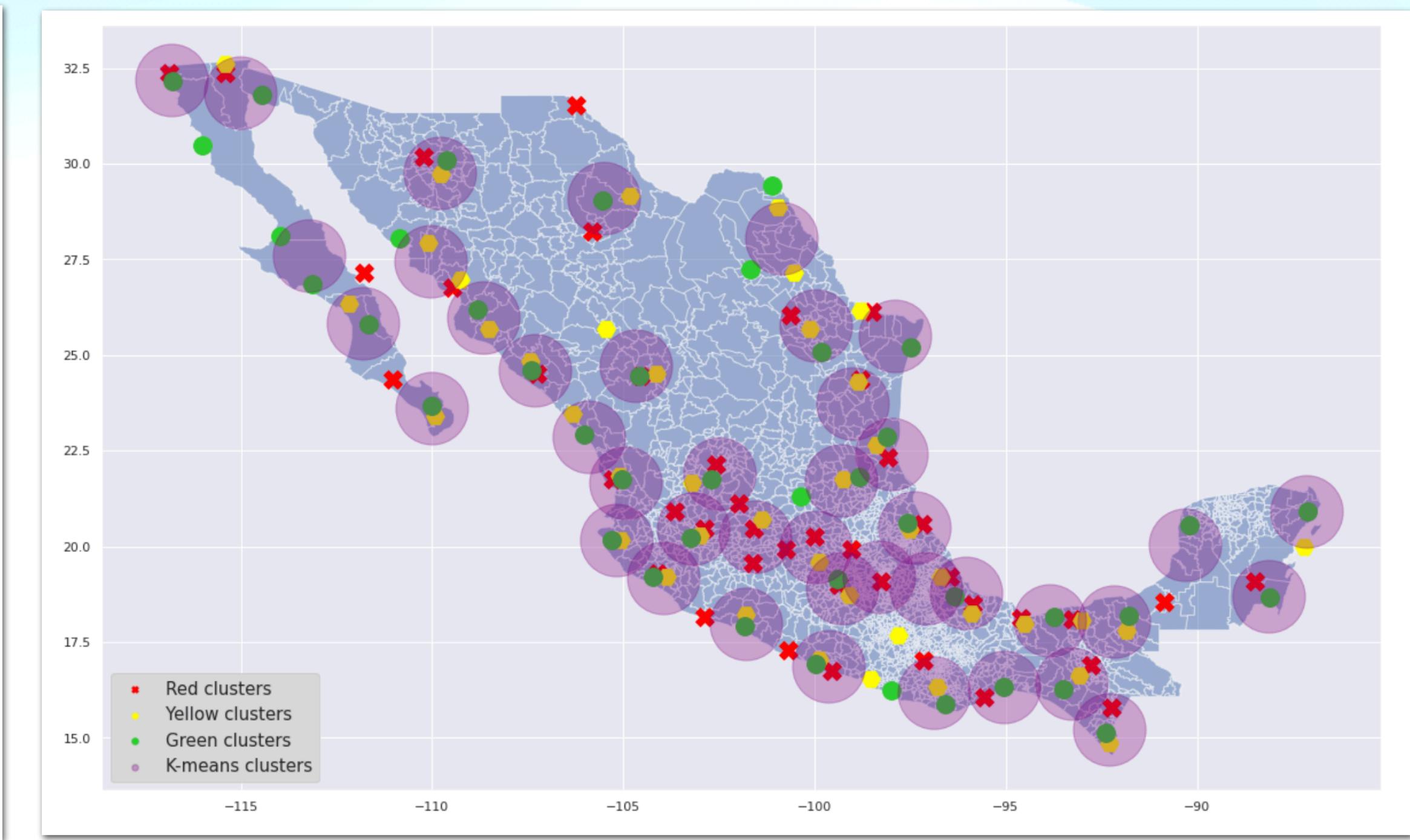
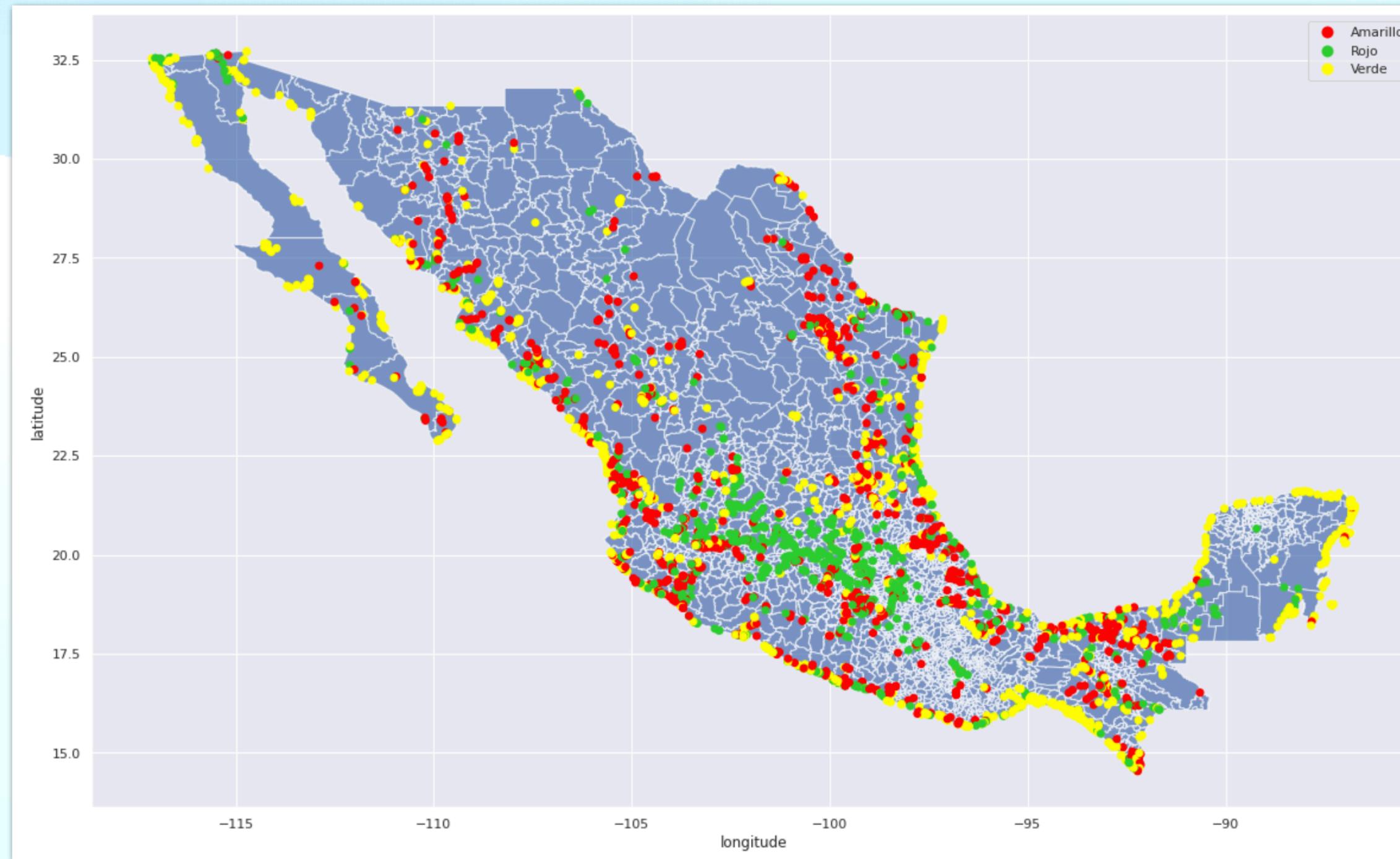
Utilizamos métodos de *elbow* y *silhouette* para determinar el número adecuado de clusters conforme al set de datos.

Conforme el análisis del codo, el número adecuado de clusters estaría ubicado alrededor de las 20 unidades. En contraste el análisis de silueta escala al canto de 40 - 50 unidades.



# K-means & Clustering

- Utilizando 41 clusters vemos como la relación con las las clases mejora, ya existen mas clusters donde predomina una clase.
- Sin embargo, parece que considerando todos los puntos del set, su ubicación geográfica y sus clases, no podríamos considerar su pertenencia a un cluster como un relación directa para la pertenencia hacia una clase.



# Clasificación

- El análisis de *features importance* mostró que el modelo de *Random Forest Classifier* distribuye la importancia en un mayor número de variables -y en una proporción mas uniforme-.

	LONGITUD	LATITUD	CALIDAD_DBO	CALIDAD_DQO	CALIDAD_SST	CALIDAD_COLI_FEC	CALIDAD_E_COLI	CALIDAD_ENTEROC	CALIDAD_OD_PORC	CALIDAD_OD_PORC_SUP	CALIDAD_OD_PORC_MED	CALIDAD_OD_PORC_FON	CALIDAD_TOX_D_48	CALIDAD_TOX_V_15	CALIDAD_TOX_D_48_SUP
0	0.034414	0.079108	0.002492	0.028605	0.012217	0.034167	0.00497	0.00398	0.002425	0.003174	0.002119	0.005968	0.001536	0.001759	0.296961
	CALIDAD_TOX_FIS_SUP_15	CONTAMINANTES	CUMPLE_CON_DBO	CUMPLE_CON_DQO	CUMPLE_CON_SST	CUMPLE_CON_CF	CUMPLE_CON_E_COLI	CUMPLE_CON_ENTEROC	CUMPLE_CON_OD	CUMPLE_CON_TOX	GRUPO				
	0.029872	0.23862	0.005003	0.069921	0.016325	0.080361	0.019822	0.0023	0.008155	0.00743	0.008296				

El modelo de *Decision Tree* sí remarcó algunas variables con una importancia notablemente mayor.

	LONGITUD	LATITUD	CALIDAD_DBO	CALIDAD_DQO	CALIDAD_SST	CALIDAD_COLI_FEC	CALIDAD_E_COLI	CALIDAD_ENTEROC	CALIDAD_OD_PORC	CALIDAD_OD_PORC_SUP	CALIDAD_OD_PORC_MED	CALIDAD_OD_PORC_FON	CALIDAD_TOX_D_48	CALIDAD_TOX_V_15	CALIDAD_TOX_D_48_SUP
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.016928e-18	0.0	0.0	0.0	0.0	0.501671
	CALIDAD_TOX_FIS_SUP_15	CONTAMINANTES	CUMPLE_CON_DBO	CUMPLE_CON_DQO	CUMPLE_CON_SST	CUMPLE_CON_CF	CUMPLE_CON_E_COLI	CUMPLE_CON_ENTEROC	CUMPLE_CON_OD	CUMPLE_CON_TOX	GRUPO				
	0.003381	0.458945	0.0	0.0	0.0	0.033735	0.0	0.002267	0.0	0.0	0.0				

- Se utilizó una proporción (80 /20) para el entrenamiento de los modelos.

```
X_cat_train, X_cat_val, y_cat_train, y_cat_val = train_test_split(X_cat, Y_cat, train_size=0.80, random_state=0)
X_con_train, X_con_val, y_con_train, y_con_val = train_test_split(X_con, Y_con, train_size=0.80, random_state=0)
```

```
crossValidation = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=0)
```

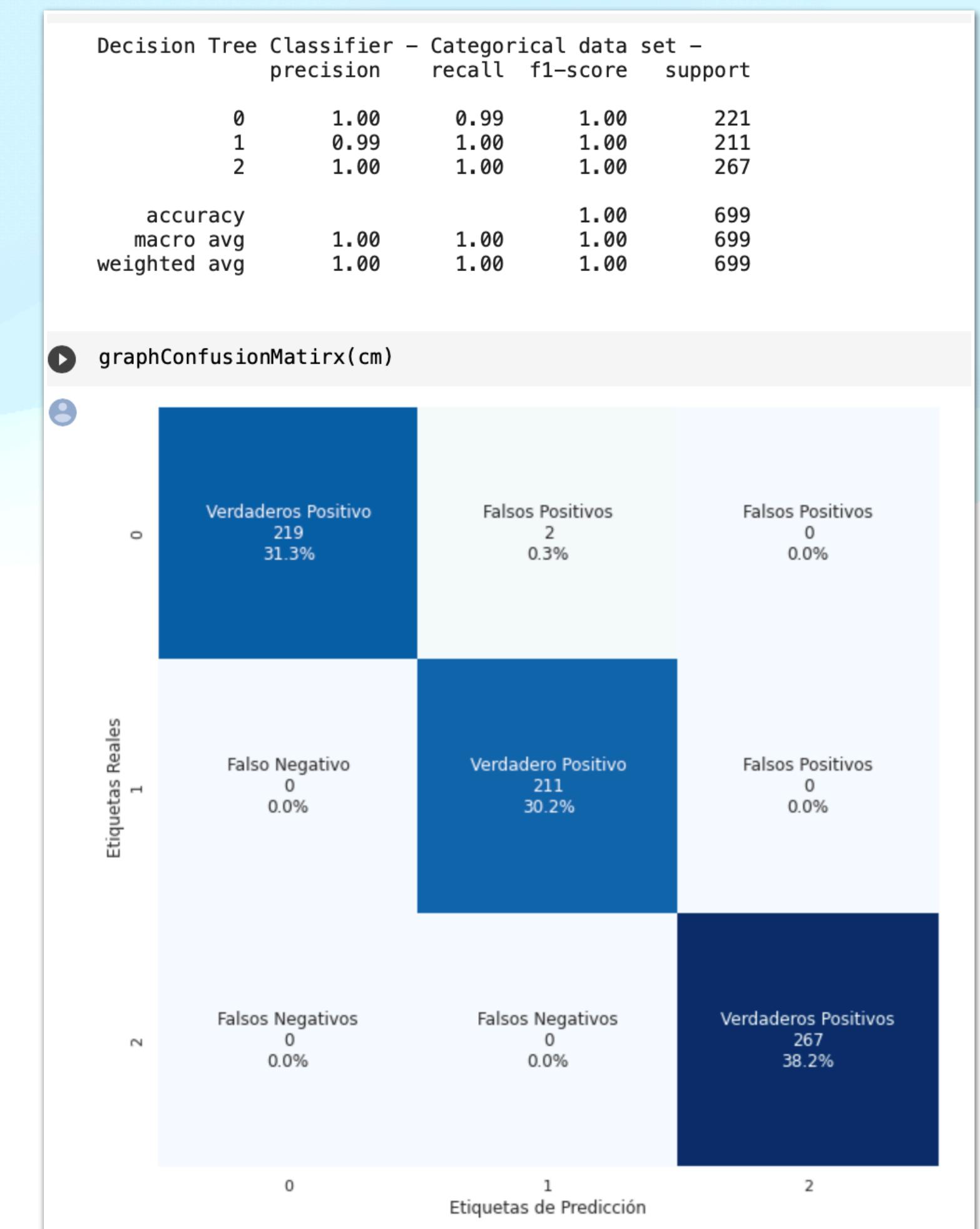
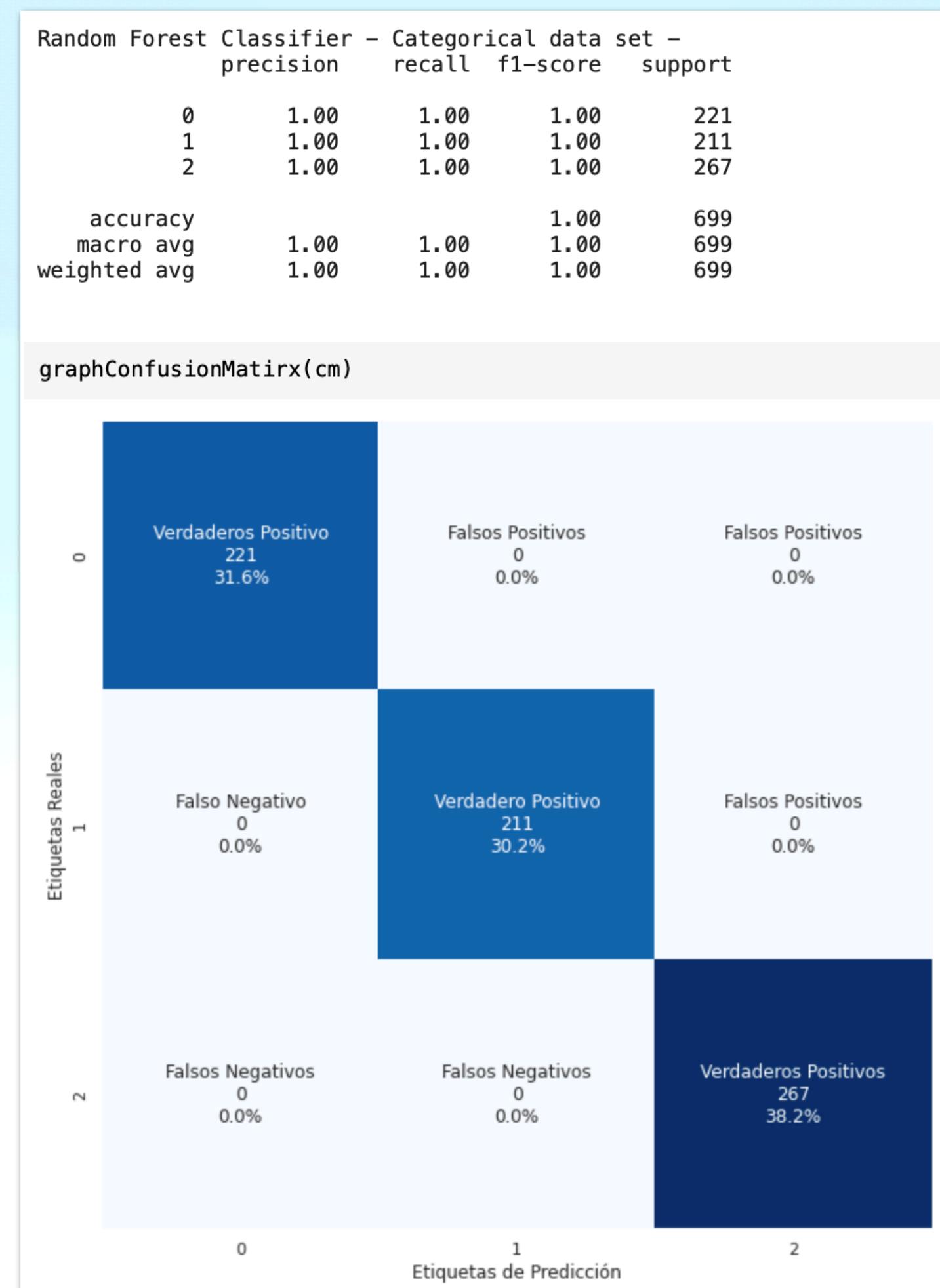
- Los resultados de entrenamiento muestran al modelo de *Random Forest Classifier* con una ligera ventaja sobre el *Decision Tree*.

```
Random Forest Classifier - Categorical data set - Test Score: 0.9977329802536105
Decision Tree Classifier - Categorical data set - Test Score: 0.9861593603529089
Random Forest Classifier - Categorical and continuous data set - Test Score: 0.9976139333978795
Decision Tree Classifier - Categorical and continuous data set - Test Score: 0.9861593603529089
```

# Resultados

- El rendimiento de los modelos es bastante similar, pero el Random Forest Classifier tiene un ventaja mínima.
- El árbol de decisión solo muestra un error de clasificación de dos muestras sobre la partición de datos de validación.

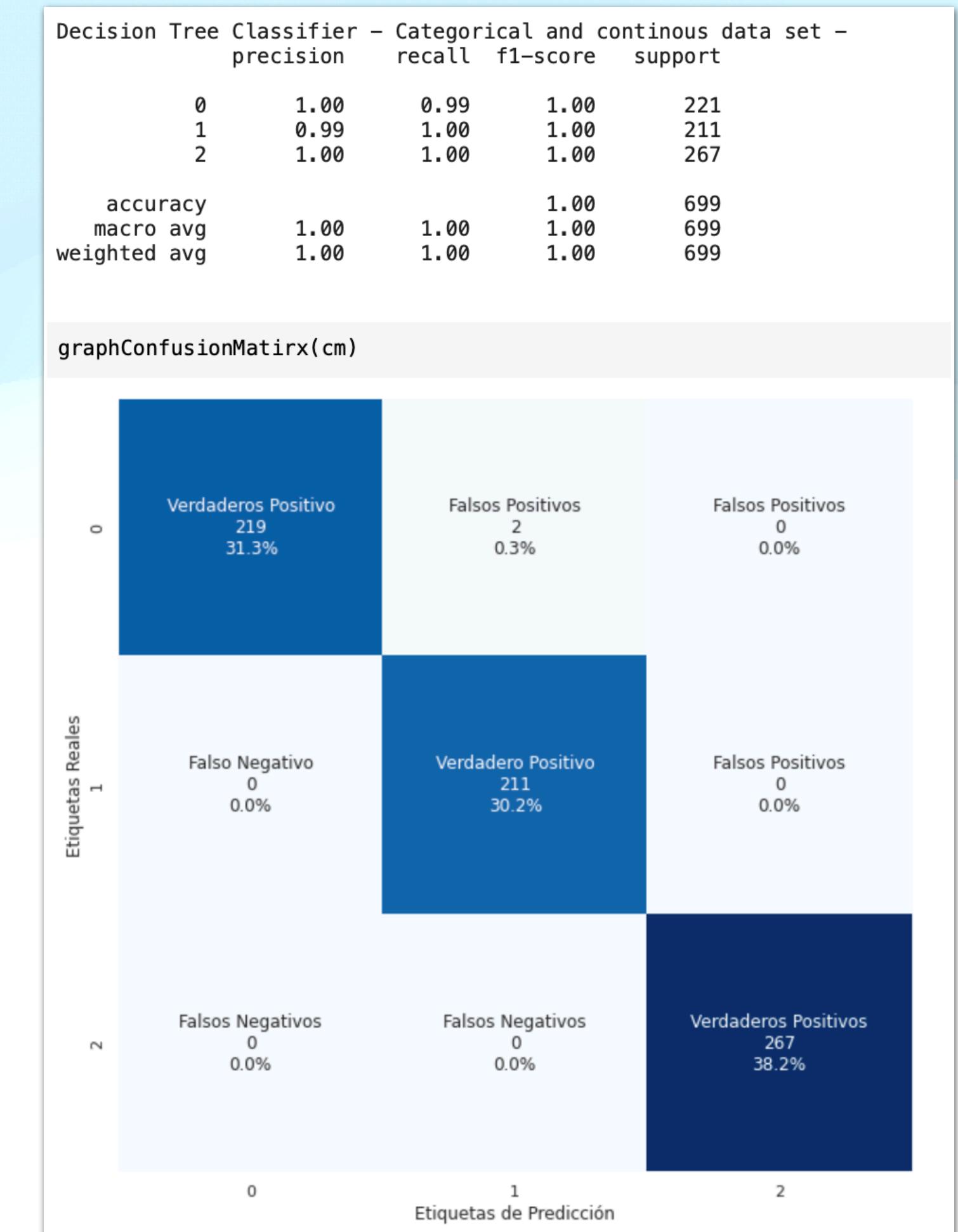
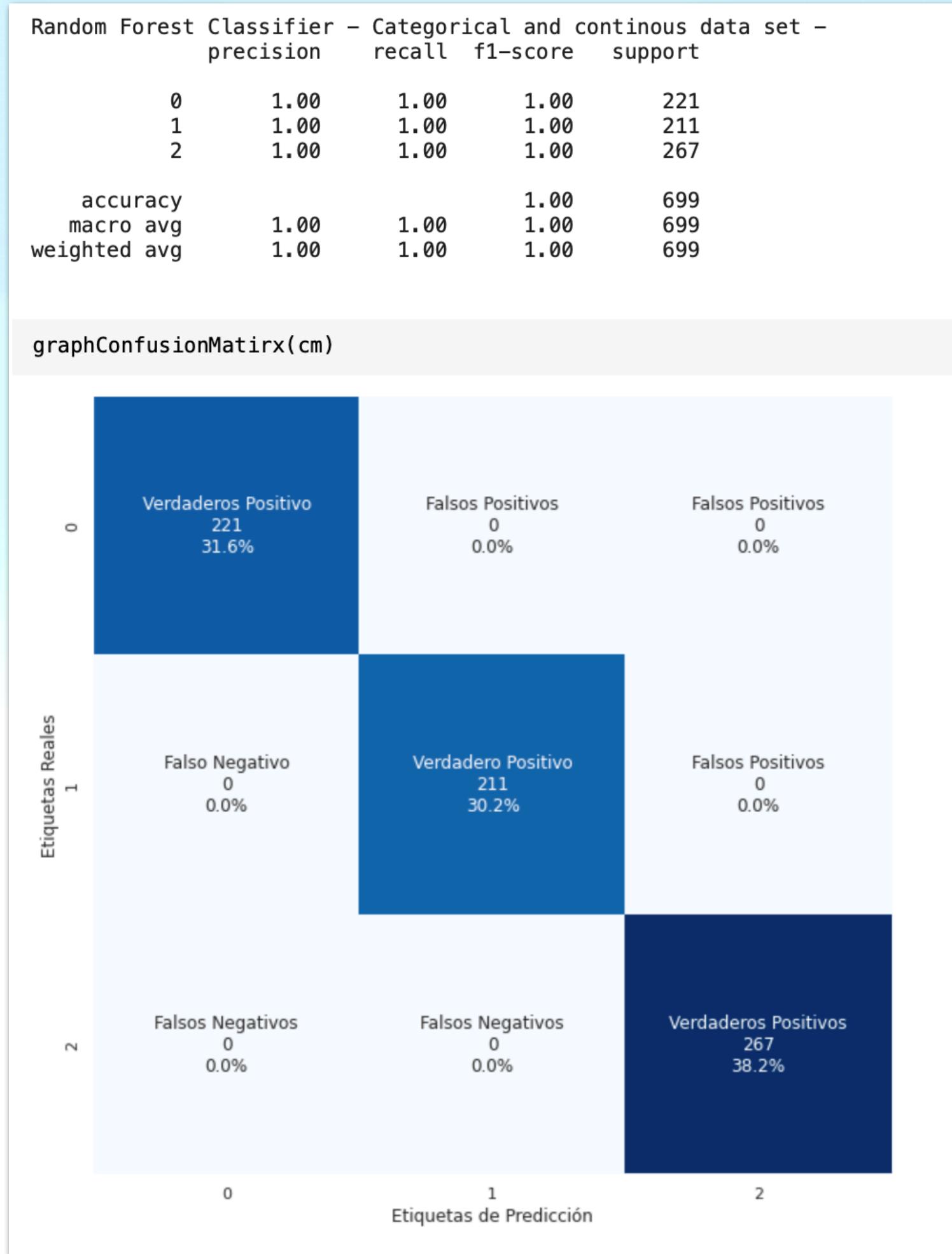
## Data Set con variables categóricas



# Resultados

Los resultados en rendimiento son consistentes cuando se usa el dataset que contiene variables continuas, incluso los errores de clasificación se presentan sobre las mismas clases.

## Data Set con variables categóricas y continuas



# Conclusiones

- Para este problema no se puede determinar una relación entre el semáforo de un cuerpo de agua y su pertenencia a un cluster -incluso con un número elevado de clusters.

Una propuesta interesante sería segmentar las muestras por regiones - por ejemplo zona centro, o en zonas costeras- observando la ubicación y concentración de los puntos en el mapa podemos pensar que la *clusterización* pudiera resultar mas efectiva.
- No se generó una diferencia significativa en el rendimiento al usar los diferentes sets de datos (exclusivamente categóricas vs categóricas y continuas), pero el análisis de *feature importance* si demostró que para el árbol de decisión podríamos considerar una reducción de dimensionalidad importante en comparación al *random forest*.
- La naturaleza de los datos, con categorías claramente definidas puede haber influido en la alta precisión de ambos modelos.