Maestría en Inteligencia Artificial Aplicada

Ciencia y análitica de datos

Dalina Aidee Villa Ocelotl (A01793258)
Miguel Guillermo Galindo Orozco (A01793695)

Actividad semanal 9: Reto parte 2

Clasificación-ensambles

Profra. María de la Paz Rico Fernández

15 de noviembre de 2022

Reto: Parte 2

Resumen

El presente trabajo considera la investigación y la aplicación de los conocimientos relacionados a la creación de un algoritmo de clasificación que nos permita distinguir la calidad del agua de acuerdo a la creación de grupos que determinen la calidad del agua subterránea. La mejora del conocimiento del recurso natural mejorará su gestión y su uso.

Introducción

En el análisis e investigación de problemas aplicados la calidad y fuente de datos es muy importante pero sobre todo el tipo de análisis y la forma en que se aborde el tratamiento de los datos, en esta segunda parte el enfoque principal es el modelado y comparación de la mejor estrategia que se debe considerar de acuerdo a los datos estudiados en la parte 1.

La importancia de este problema radica en que el conocimiento de las características generales es la correcta aplicación de modelos de aprendizaje supervisados para conocer la correcta clasificación de los acuíferos con mejor calidad del agua.

Contexto de aplicación (También se incluye en Parte 1)

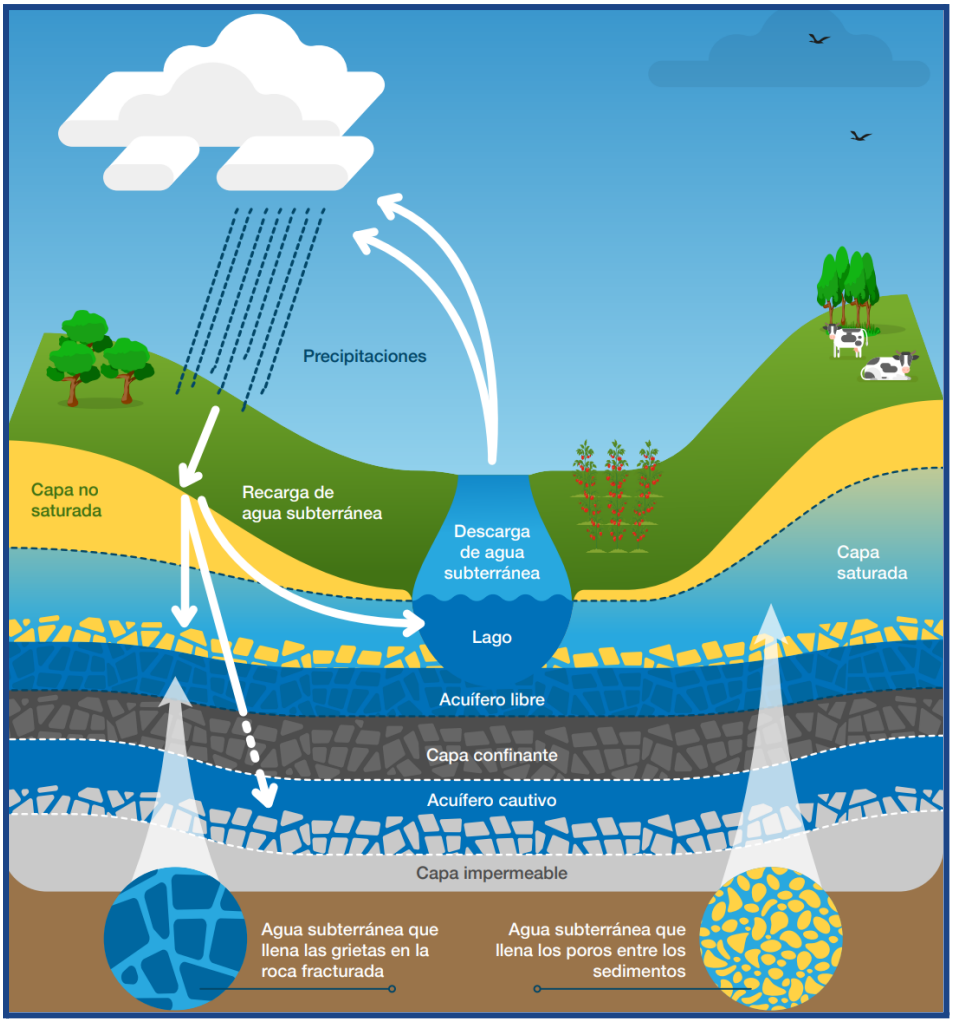
Como en toda aplicación es importante el contexto de aplicación acerca del tipo de aguas, en este caso se decidió estudiar aguas subterráneas.

Aguas subterráneas

Las aguas subterráneas son aquellas que habitan debajo de la tierra, tan solo en México se tienen identificados 653 acuíferos y el 38.7% proviene de esas fuentes por lo que su explotación es importante para satisfacer las necesidades de la sociedad mexicana actual.

Una de las principales características es que proviene del agua de lluvia y que constantemente se encuentra en un proceso de limpia y purificación, denominado infiltración de la lluvia, este proceso se desarrolla de manera más lenta que encima de la superficie. Esta agua se almacena de forma natural en depósitos debajo de la superficie terrestre, conocidos como mantos acuíferos subterráneos,

Una observación importante es el constante flujo del agua subterránea, eso se debe a la porosidad y permeabilidad de los mantos acuíferos y los ríos que los conectan entre ellos, que se expanden por todas las cuencas hidrográficas completas.



Calidad del agua subterránea

Se refiere a la temperatura del agua, la cantidad de sólidos disueltos y a la ausencia de contaminantes tóxicos y biológicos, para ellos es necesario conocer sus condiciones fisicas, quimicas y microbiologicas, donde las concentraciones de agentes tóxicos se dan por aportes externos y no a condiciones naturales.

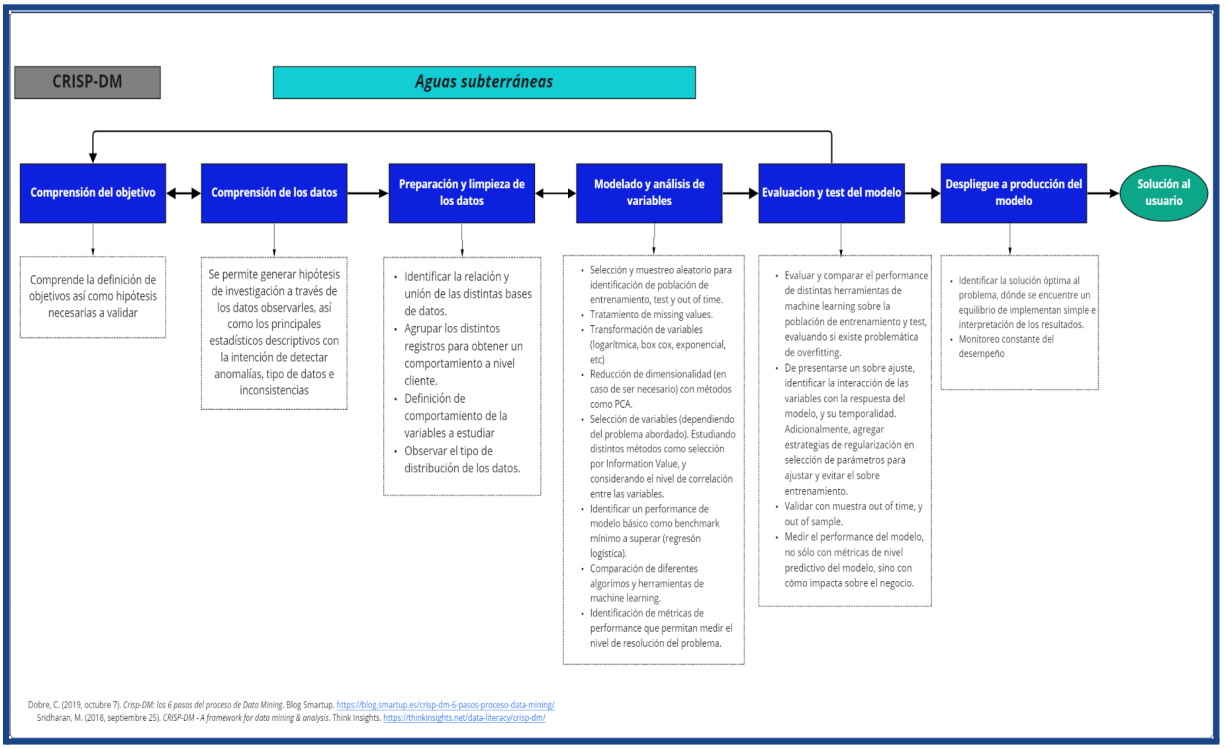
Hidrogeoquímica y calidad del agua subterránea

Tienen su origen en la procedencia de sus iones, provenientes de la disolución de las diferentes formaciones geológicas, se clasificó las familias de agua predominantes, teniendo que en la mayor parte la zona predomina la sódica Clorurada, agua característica de la intrusión por salinidad marina. Hacia la parte poniente y sur de la zona se localizan zonas de familias mixta sódica Clorurada y mixta Cálcica Clorurada.

Metodología de investigación

De acuerdo a las metodologías con las que se puede abordar la investigación, se plantea la metodología de CRISP-DM (Cross-Industry Standard Process for Data Mining), contiene principalmente 6 fases aunque estas pueden ajustarse de acuerdo a la investigación. La importancia de aplicar una metodología reconocida es porque favorece el proceso de

investigación científica, además de que un método se necesita para ordenar, esquematizar, registrar e interpretar datos.



https://miro.com/app/board/uXjVPTib1mw=/

Desarrollo de la investigación (Continuación Parte 1)

Modelado y análisis de las variables:

Se tiene una base de datos de información de acuíferos en México su entendimiento es primordial porque garantiza el uso adecuado de los recursos naturales.

La liga del desarrollo del modelado de la información:

https://github.com/PosgradoMNA/actividades-del-projecto-equipo_88/blob/main/Reto_Eq_88_parte_2.ipynb

Selecciona tus variables independientes (X) y dependiente (Y- Semáforo)

Se define la variable Semáforo como nuestra variable target sobre la cual se deben tomar las decisiones de entrenamiento de la calidad del agua.

```
X = df.drop(columns=['CLAVE','PERIODO','SEMAFORO'])

y = df[['SEMAFORO']]
```

```
SEMAFORO
Amarillo    247
Rojo        387
Verde       434
dtype: int64
```

Dividir los datos de manera balanceada y definición de pipeline (One Hot Encoding para categóricas)

La importancia de los modelos radica en que puedan predecir de manera acertada, para lograr esto se debe asegurar que el modelo tenga capacidad de pronóstico y esto se consigue separando la base de datos.

```
Xtv, Xtest, ytv, ytest = train_test_split(X, y_ohe, test_size=0.2, random_state=1234)
```

(854, 53) : dimensión de datos de entrada para entrenamiento y validación

(214, 53) : dimensión de datos de entrada para prueba

(854, 1) : dimensión de variable de salida para entrenamiento y validación

(214, 1) : dimensión de variable de salida para prueba

Transformar las otras variables categóricas por One Hot Encoding

Se realiza la unión de las transformaciones numéricas y categóricas que se estarán aplicando a los datos de entrada:

```
columnasTransformer = ColumnTransformer(transformers = [('catohe', catOHE_pipeline, catOHE_pipeline_nombres)], remainder='passthrough')
```

Realiza un análisis general de las features importances a través de “decision trees”

Se realiza un único árbol de decisión con profundidad alta para observar qué features toma cómo los más importantes, y hacer una selección de atributos. Es la creación de un esqueleto del modelo de predicción que representa un árbol donde las ramas constituyen los patrones reconocidos en el proceso de aprendizaje.

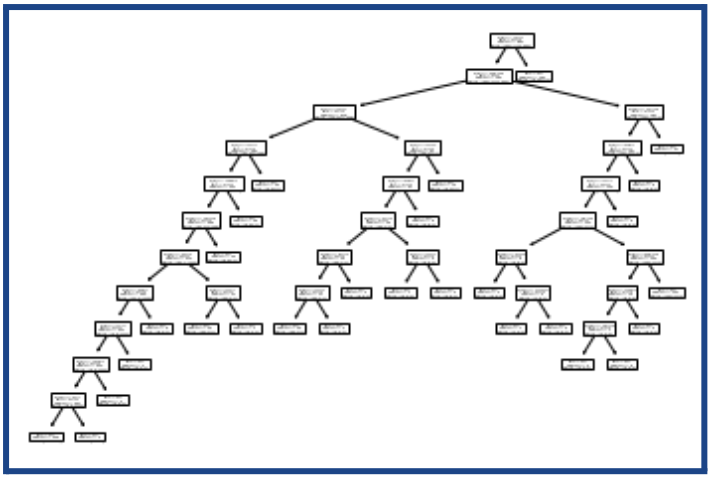
Se utilizan sólo los atributos numéricos en el set de entrenamiento, para seleccionar aquellos con mayor importancia.

Se decide que se utilizará el set completo de atributos categóricos, ya que en ellos se resume bastante información con impacto en la variable de semáforo.

```
Tree = DecisionTreeClassifier(max_depth=30)
```

```
DecisionTreeClassifier(max_depth=30)
```

Feature Importances



Selecciona las variables de mayor importancia.

Decidimos tomar el top 5 variables numéricas por el feature importance mostrado.

La característica principal es que las primeras 5 representan cada uno al menos 10% de importancia en el modelo de sólo variables numéricas.

Con esto las variables seleccionadas son:

- FLUORUROS_mg
- DUR_mg
- N_NO3_mg
- COLI_FEC_NMP
- AS_TOT_mg

Explorar clasificador óptimo

Decidimos probar los siguientes clasificadores:

- Regresión Logística (Benchmark): es el clasificador base, en la industria se utiliza como el mínimo posible a superar.
- Árbol de decisión: simpleza en la decisión de clasificación, tomando sólo los atributos más importantes.
- Random Forest: ensamble de árboles que permite entrenar con distintos atributos en cada árbol, evitando sobre ajustar, y minimizar el error.

En cada método se decidió hacer una búsqueda de malla (Grid Search), sobre un set de hiper parámetros que permita encontrar la mejor combinación de ellos para optimizar el resultado. Adicionalmente, se hace uso de validación cruzada por muestras aleatorias (RepeatedStratifiedKFold) para evitar el sobre ajuste, y optimizar el resultado, disminuyendo el error.

Regresión Logística

```
log_modelo = LogisticRegression(solver='liblinear',max_iter=5000)
```

Búsqueda de la mejor regresión logística como clasificador:

- Grid search de parámetros.
- Método de validación cruzada de muestras para evitar sobre ajuste, y sub ajuste.

Validación cruzada de 5 muestras con 3 repeticiones

```
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=1234)
```

Se hace la búsqueda de malla dentro del rango de los hiperparámetros propuestos, y sobre las validaciones cruzadas propuestas en el objeto (RepeatedStratifiedKFold)

```
log_grid =GridSearchCV(estimator=log_modelo, param_grid=dicc_hiperparam, cv=cv, scoring='accuracy')
```

Transformación de datos categóricos conforme al pipeline definido anteriormente.

variables_seleccionadas = seleccion_num + catOHE_pipeline_nombres

```
Xtv_seleccion = Xtv[variables_seleccionadas]
```

```
Xx = columnasTransformer.fit_transform(Xtv_seleccion)
```

Se entrena el modelo sólo sobre la población de train

```
GridSearchCV(cv=RepeatedStratifiedKFold(n_repeats=3, n_splits=5, random_state=1234),
             estimator=LogisticRegression(max_iter=5000, solver='liblinear'),
                       param_grid= {'C': [1e-05, 10.0, 10000.0], 'class_weight': ['balanced', None], 'penalty': ['l1', 'l2'], 'solver': ['liblinear']},  scoring='accuracy')
```

Visualización de resultados obtenidos:

- Exactitud (accuracy) encontrado con el método de optimización.
- Hiperparámetros utilizados que optimizan el accuracy.

Mejor valor de exactitud obtenido con la mejor combinación: 0.9945373237014103

Mejor combinación de valores encontrados de los hiperparámetros: {'C': 10.0, 'class_weight': None, 'penalty': 'l1', 'solver': 'liblinear'}

Los resultados obtenidos son sospechosamente altos, y se sugiere que el modelo va a sobre ajustar sobre una población no conocida.

```
log_model = LogisticRegression(C=10.0, class_weight=None, penalty='l1', solver='liblinear')
```

La exactitud obtenida con la mejor combinación en el set de validación (test) es: 0.9953271028037384

Se decide no utilizar este modelo a pesar de los resultados favorable, ya que es posible que sobre ajuste en población no conocida

Se sugiere realizar pruebas out of time, y out of sample

Árbol de Decisión

Búsqueda del mejor árbol:

- Grid search de parámetros.
- Método de validación cruzada de muestras para evitar sobre ajuste, y sub ajuste.

validación cruzada de 5 muestras con 3 repeticiones

```
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=1234)
```

Transformación de datos categóricos conforme al pipeline definido anteriormente.

```
Xtv_seleccion = Xtv[variables_seleccionadas]

Xx = columnasTransformer.fit_transform(Xtv_seleccion)
```

Se entrena el modelo sólo sobre la población de train

```
dt_grid.fit(Xx, ytv)

GridSearchCV(cv=RepeatedStratifiedKFold(n_repeats=3, n_splits=5, random_state=1234),

             estimator=DecisionTreeClassifier(),

                          param_grid={'ccp_alpha': [0.1, 0.01],'class_weight': ['balanced', None],
'criterion': ['gini', 'entropy'], 'max_depth': array([1, 2]), 'min_samples_split': array([2, 3, 4, 5, 6, 7,
8, 9])}, scoring='accuracy')
```

Visualización de resultados obtenidos:

- Exactitud (accuracy) encontrado con el método de optimización.
- Hiperparámetros utilizados que optimizan el accuracy.

Mejor valor de exactitud obtenido con la mejor combinación: 0.8364407751404656

Mejor combinación de valores encontrados de los hiperparámetros: {'ccp_alpha': 0.01, 'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 2, 'min_samples_split': 2}

El performance del modelo con mejores hiperparámetros en validación:

```
modeloDT = DecisionTreeClassifier(ccp_alpha=0.01, class_weight='balanced', criterion='gini',
max_depth=2, min_samples_split=2)
```

La exactitud obtenida con la mejor combinación en el set de validación (test) es: 0.8130841121495327

Random Forest

```
modeloRF = RandomForestClassifier()
```

Búsqueda del mejor ensamble de árboles por la metodología de Random Forest:

- Grid search de parámetros.
- Método de validación cruzada de muestras para evitar sobre ajuste, y sub ajuste.

Se incluyen los siguientes hiperparametros a la búsqueda: ccp_alpha, criterion, max_depth, min_samples_split y class_weight.

Para el mejor ensamble de árboles con Random Forest

```
rf_param_grid = {'ccp_alpha': [0.1, .01,0.001], 'criterion': ['gini', 'entropy'], 'max_depth': np.arange(1, 3, step=1), 'min_samples_split': np.arange(2, 10, step=1), 'class_weight': ['balanced', None]}
```

Validación cruzada de 5 muestras con 3 repeticiones

```
cv = RepeatedStratifiedKFold(n_splits=5, n_repeats=3, random_state=8)
```

Transformación de datos categóricos conforme al pipeline definido anteriormente.

```
Xtv_seleccion = Xtv[variables_seleccionadas]
```

```
Xx = columnasTransformer.fit_transform(Xtv_seleccion)
```

Se entrena el modelo sólo sobre la población de train

```
rf_grid.fit(Xx, ytv.ravel())
```

```
GridSearchCV(cv=RepeatedStratifiedKFold(n_repeats=3,n_splits=5, random_state=8),estimator=RandomForestClassifier(), param_grid= {'ccp_alpha': [0.1, 0.01, 0.001], 'criterion': ['gini', 'entropy'], 'max_depth': array([1, 2]), 'min_samples_split': array([2, 3, 4, 5, 6, 7, 8, 9])}, scoring='accuracy')
```

Visualización de resultados obtenidos:

- Exactitud (accuracy) encontrado con el método de optimización.
- Hiperparámetros utilizados que optimizan el accuracy.

Mejor valor de exactitud obtenido con la mejor combinación: 0.893078775369797

Mejor combinación de valores encontrados de los hiperparámetros: {'ccp_alpha': 0.01, 'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 2, 'min_samples_split': 2}

El performance del modelo con mejores hiper parámetros en validación:

modeloDT = RandomForestClassifier(ccp_alpha=0.001, class_weight='balanced', criterion='gini', max_depth=2, min_samples_split=3)

La exactitud obtenida con la mejor combinación en el set de validación (test) es: 0.8457943925233645

Clasificador seleccionado

Con los resultados obtenidos elegimos Random Forest por su poder de clasificación con un Accuracy en Train de 0.89 y en Test de 0.85

Adicionalmente, por la naturaleza del algoritmo se espera que no sobre ajuste en el TimestampConvertibleTypes

El modelo propuesta con sus resultados serían los siguientes:

modelo_final = RandomForestClassifier(ccp_alpha=0.001, class_weight='balanced', criterion='gini', max_depth=2, min_samples_split=3)

La exactitud en el accuracy obtenida con la mejor combinación en el set de Entrenamiento es: 0.8992974238875878

La exactitud obtenida con la mejor combinación en el set de validación (test) es: 0.8598130841121495

Reporte de clasificación

	precision	recall	f1-score	support
Amarillo	0.76	0.98	0.86	43
Rojo	0.98	0.63	0.77	78
Verde	0.85	1.00	0.92	93
accuracy			0.86	214
macro avg	0.87	0.87	0.85	214
weighted avg	0.88	0.86	0.85	214

Gráfica de precisión recall

Al ser un problema multiclase, se entrena el modelo de nuevo con las especificaciones para poder realizar el plot por clase.

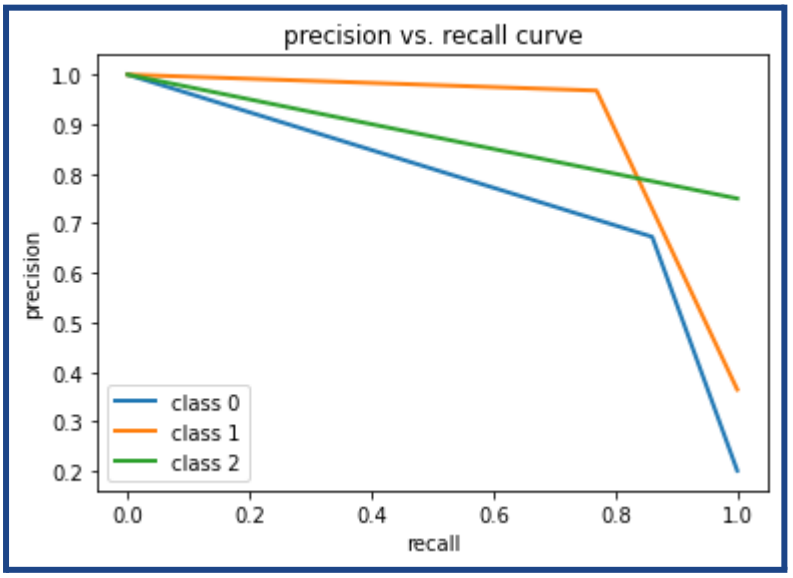
Definimos la precisión es el ratio o porcentaje de clasificaciones correctas de nuestro clasificador. En otras palabras, de todo lo que nuestro clasificador clasifica como positivo, correcta o incorrectamente (TP + FP), cual es el ratio de clasificaciones correctas.

$$precision = \frac{TP}{TP + FP}$$

Por otro lado, el recall o sensibilidad de nuestro modelo es el ratio de positivos detectado en el dataset por nuestro clasificador. En otras palabras, de todos los positivos reales de nuestro dataset, detectados o no(TP + FN), cual es el ratio de positivos detectados.

$$recall = \frac{TP}{TP + FN}$$

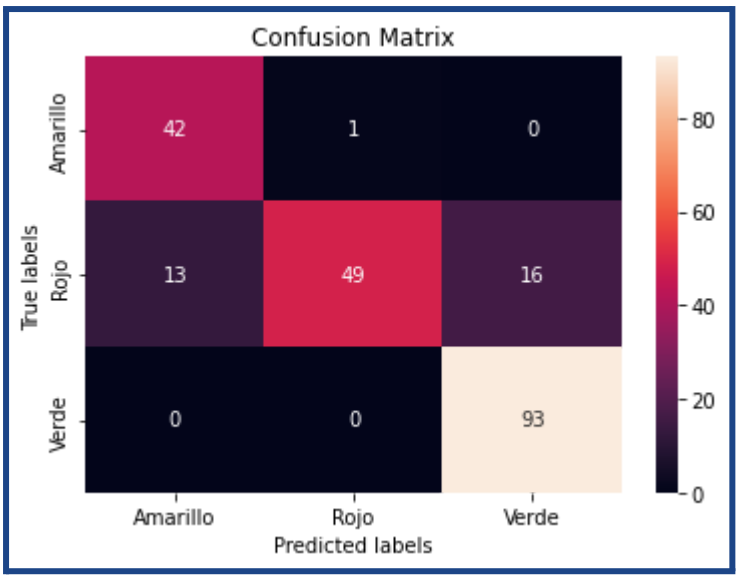
Al evaluar los datos, nos muestra los siguientes resultados:



Alta precisión y alto recall nos dice que el modelo de Machine Learning escogido maneja perfectamente esa clase 2. Por otro lado, la baja precisión y alto recall nos indica que ell modelo de Machine Learning escogido detecta bien la clase, pero también incluye muestras de otra clase 0 y clase 1.

Matriz de confusión en Set validación del modelo seleccionado

El comparativo en que muestran los valores reales comparados con los de predicción, por ellos se observa el comparativo de las métricas para identificar cuales son los resultados correctos.



Conclusiones

Se observó durante todo el desarrollo de la metodología de investigación la importancia de la calidad de los datos, en específico en la variable de semáforo de target nos permite identificar la calidad del agua adecuada. Sin embargo, observamos que el semáforo depende de las variables que entre ellas se observaba cierta correlación.

Finalmente los resultados del modelo final, nos muestran que es una buena predicción siendo el grupo rojo el más complicado de clasificar, ya que es donde se encuentran distintos niveles de los componentes en el agua. Las características del semáforo verde son bastantes claras y nos permite clasificar mejor el agua potable.

Esta investigacion de aplicacion analitica nos permitio conocer la importancia de lso datos y su uso para buenos fines, estos algoritmos nos permiten crear soluciones que permitan optimizar el uso de los recursos naturales, incluso generar políticas que protejan suelos y acuíferos en el país, la conciencia nace de darse cuenta de los resultados, por que se confirma que el desarrollo sustentable es fundamental del desarrollo humano.

Referencias

- Dobre, C. (2019, octubre 7). *Crisp-DM: los 6 pasos del proceso de Data Mining*. Blog Smartup. https://blog.smartup.es/crisp-dm-6-pasos-proceso-data-mining/
- Sridharan, M. (2018, septiembre 25). *CRISP-DM - A framework for data mining & analysis*. Think Insights. https://thinkinsights.net/data-literacy/crisp-dm/
- Roberto. (2020, marzo 21). *¿Cómo Construir un Buen Modelo de Machine Learning? [Parte 2] (Definiciones)*. Aprende Ingeniería; Aprende-Ingeniería. https://aprendeingenieria.com/como-construir-un-buen-modelo-de-machine-learning-parte-2-definiciones/