Siniestros Viales

Colaboradores	Nelson Andres Quiroga Ruiz
Versión	0.0.1
Fecha de revisión	v0: 16/julio/2024

1-Se ubicó la ruta respectiva del archivo

```
# dataini = pd.read_csv('historico_siniestros_bogota_d.c_-.csv')
dataini = pd.read_csv('/content/drive/MyDrive/Trabajos U/Seminario BigData/historico_siniestros_bogota_d.c_-.csv')
```

2-

A la columna llamada X se le agregó ÿ

```
[44] data = dataini

data = data.drop(['ÿX','Y','OBJECTID','FORMULARIO','CODIGO_ACCIDENTE','FECHA_OCURRENCIA_ACC','ANO_OCURRENCIA_ACC','DIRECCION','CIV','PK_CALZADA', ], axis=1)
data.head()
```

	GRAVEDAD	CLASE_ACC	LOCALIDAD	FECHA_HORA_ACC	LATITUD	LONGITUD
0	SOLO DANOS	CHOQUE	ENGATIVA	2017/06/12 05:30:00+00	4.693807	-74.090924
1	CON HERIDOS	OTRO	PUENTE ARANDA	2020/11/19 02:05:00+00	4.603000	-74.121000
2	SOLO DANOS	CHOQUE	USAQUEN	2020/11/10 13:30:00+00	4.682000	-74.042000
3	SOLO DANOS	CHOQUE	CIUDAD BOLIVAR	2015/05/11 10:50:00+00	4.587187	-74.166937
4	SOLO DANOS	CHOQUE	LOS MARTIRES	2016/06/08 21:30:00+00	4.607648	-74.092901

Partición conjuntos de datos: Entrenamiento, prueba y validación, se quitó el comentario en el segundo conjunto de comandos para ejecutar redes neuronales

```
[65] from sklearn.model_selection import train_test_split as split
```

3-

```
[74] ## # Definir función para particionar el conjunto de datos

     X_train, X_test, y_train, y_test = split(Xt,Y,
                                              test_size=0.30,
                                              random_state=42,
                                              stratify=Y)

     X_train, X_valid, y_train, y_valid = split(X_train, y_train,
                                                test_size=0.2,
                                                stratify=y_train)
```

4-

Modelos Supervisados

Árboles de decisión, se quitó el comentario en la primera linea

```
import xgboost as xgb
from sklearn import metrics
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Primer modelo supervisado resultados:

	precision	recall	f1-score	support
0	0.53	0.54	0.53	20310
1	0.07	0.08	0.07	972
2	0.76	0.75	0.76	38462
accuracy			0.67	59744
macro avg	0.45	0.46	0.46	59744
weighted avg	0.67	0.67	0.67	59744

Segundo modelo Random forest

	precision	recall	f1-score	support
0	0.79	0.45	0.57	20310
1	0.00	0.00	0.00	972
2	0.76	0.95	0.84	38462
accuracy			0.76	59744
macro avg	0.51	0.47	0.47	59744
weighted avg	0.76	0.76	0.74	59744

Redes neuronales

Redes Neuronales Artificiales

Se añadió !pip install tersorflow

```
[92] !pip install tensorflow
```

Se añadió líneas de código para visualizar la cantidad de filas y columnas de los datos de entrenamiento.

```
# Entrenar la red neuronal

print("Shape of X_train:", X_train.shape)
print("Shape of Y_train:", Y_train.shape)
print("Shape of X_valid:", X_valid.shape)
print("Shape of Y_valid:", Y_valid.shape)

history_rna = trainrna(model_rna, 5, 32, X_train,Y_train,X_valid,Y_valid)
```

```
Shape of X_train: (111521, 33)
Shape of Y_train: (111521, 1)
Shape of X_valid: (27881, 33)
Shape of Y_valid: (27881, 1)
Epoch 1/5
 843/3486 [======>.....................] - ETA: 14s
```

La línea de código produjo el error:

```
ValueError                                Traceback (most recent call last)
<ipython-input-106-e3448b6cdbaf> in <cell line: 8>()
      6 print("Shape of Y_valid:", Y_valid.shape)
      7
----> 8 history_rna = trainrna(model_rna, 5, 33, X_train,Y_train,X_valid,Y_valid)

                                  2 frames
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py in fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split, validation_data, shuffle, class_weight,
sample_weight, initial_epoch, steps_per_epoch, validation_steps, validation_batch_size, validation_freq, max_queue_size, workers, use_multiprocessing)
   1817                 logs = tf_utils.sync_to_numpy_or_python_type(logs)
   1818               if logs is None:
-> 1819                   raise ValueError(
   1820                       "Unexpected result of `train_function` "
   1821                       "(Empty logs). This could be due to issues in input "

ValueError: Unexpected result of `train_function` (Empty logs). This could be due to issues in input pipeline that resulted in an empty dataset. Otherwise, please use `Model.compile(...,
run_eagerly=True)`, or `tf.config.run_functions_eagerly(True)` for more information of where went wrong, or file a issue/bug to `tf.keras`.
```

Se agregó una función para compilar el modelo y optimizarlo antes de pasarlo a la función de entrenamiento “trainrna()”.

```
# Entrenar la red neuronal

print("Shape of X_train:", X_train.shape)
print("Shape of Y_train:", Y_train.shape)
print("Shape of X_valid:", X_valid.shape)
print("Shape of Y_valid:", Y_valid.shape)
model_rna.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history_rna = trainrna(model_rna, 5, 33, X_train,Y_train,X_valid,Y_valid)
```

```
Shape of X_train: (111521, 33)
Shape of Y_train: (111521, 1)
Shape of X_valid: (27881, 33)
Shape of Y_valid: (27881, 1)
Epoch 1/5
3380/3380 [==============================] - 11s 3ms/step - loss: 0.5972 - accuracy: 0.7583 - val_loss: 0.5465 - val_accuracy: 0.7708
Epoch 2/5
3380/3380 [==============================] - 7s 2ms/step - loss: 0.5428 - accuracy: 0.7706 - val_loss: 0.5429 - val_accuracy: 0.7706
Epoch 3/5
3380/3380 [==============================] - 9s 3ms/step - loss: 0.5405 - accuracy: 0.7711 - val_loss: 0.5416 - val_accuracy: 0.7709
Epoch 4/5
3380/3380 [==============================] - 7s 2ms/step - loss: 0.5392 - accuracy: 0.7711 - val_loss: 0.5403 - val_accuracy: 0.7710
Epoch 5/5
3380/3380 [==============================] - 9s 3ms/step - loss: 0.5381 - accuracy: 0.7716 - val_loss: 0.5398 - val_accuracy: 0.7712
```

Se obtuvo:

```
precision: [0.89863791 0.         0.74796301]
recall: [0.38980798 0.         0.99048411]
fscore: [0.54375    0.         0.85230718]
support: [20310   972 38462]
```

(Hasta ahora el modelo con mejor precisión)

Matriz de confusión:

```
1867/1867 [==============================] - 2s 1ms/step
Confusion Matrix
[[ 8810     1 50933]
 [    0     0     0]
 [    0     0     0]]
```

Xgboost

```
Mean cross-validation score: 0.77

              precision    recall  f1-score   support

           0       0.88      0.41      0.56     20310
           1       0.00      0.00      0.00       972
           2       0.75      0.98      0.85     38462

    accuracy                           0.77     59744
   macro avg       0.54      0.46      0.47     59744
weighted avg       0.78      0.77      0.74     59744
```

Conclusión: El método más preciso en nuestro análisis resultaron ser las redes neuronales implementadas con TensorFlow. Esto se debe a que, al evaluar las métricas de rendimiento, las redes neuronales mostraron un rendimiento superior en términos de precisión predictiva comparado con los demás, aunque las redes neuronales ofrecen una alta precisión, también requieren más recursos computacionales y tiempo de entrenamiento