Variable "Tipologia"

Cargamos las librerias necesarias para la elaboracion y desarrollo de la minería de datos

```
!pip install plotly
import numpy as np #Operaciones matemáticas rápidas sobre matrices
import pandas as pd #biblioteca de análisis y manipulación de datos para Python
import plotly.express as px
import matplotlib.pyplot as plt #Proporciona una forma de trazado similar a MATLAB. pyplot está diseñado principalmente para gráficos in
import seaborn as sns #permite generar fácilmente elegantes gráficos, proporciona una interfaz de alto nivel que es realmente sencilla d
import statsmodels.api as sm

# Preprocesado y modelado
# ------------------------------------------------------------------------------
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.tree import export_graphviz
from sklearn.tree import export_text
from sklearn.model_selection import GridSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

# Configuración warnings
# ------------------------------------------------------------------------------
import warnings
warnings.filterwarnings('once')
#Dataset
# ------------------------------------------------------------------------------
df= pd.read_csv('AT2021_NBD.csv')
df.head()
```

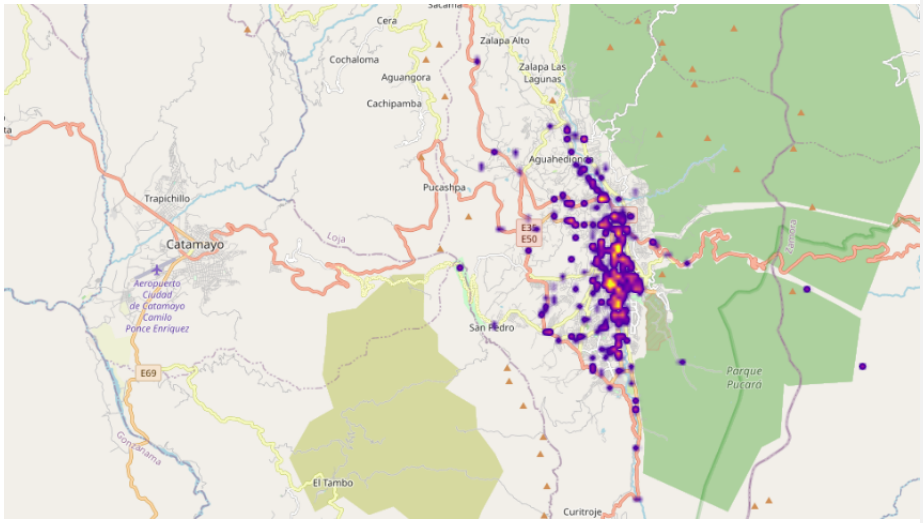
```
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages (5.13.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly)
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; fa
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling bac
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; fa
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling bac
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; fa
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling bac
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back
```

	fecha	dia	hora	latitud	longitud	tipologia	direccion	zona	parroquia_urbana	cau
0	01-01	viernes	h11	-3.991993	-79.201155	estrellamiento	orillas del zamora y jose felix de valdivieso	urbana	el valle	conduci estad embriag
1	01-05	martes	h17	-4.020370	-79.217962	choque lateral perpendicular	benjamin carrion y gustavo serrano	urbana	punzara	conduci estad embriag
2	01-05	martes	h12	-3.987230	-79.202984	choque por alcance	nueva loja y guaranda	urbana	sucre	no mant la dista reglament
3	01-08	viernes	h10	-3.989410	-79.236506	atropello	angel felicisimo rojas	urbana	sucre	imprude del pe
4	01-09	sabado	h06	-3.979784	-79.218689	choque lateral angular	isidro ayora y habana	urbana	sucre	conduci estad embriag

Graficamos el mapa de calor de accidentabilidad dentro del cantón Loja

```
fig = px.density_mapbox(df,lat='latitud', lon='longitud',radius=3,center=dict(lat=-3.99313,lon=-79.20422),zoom=10.5,mapbox_style="open-s

fig.show()
```

```
<frozen importlib._bootstrap>:914: ImportWarning:

APICoreClientInfoImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_PyDriveImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_OpenCVImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_BokehImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_AltairImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

APICoreClientInfoImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_PyDriveImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_OpenCVImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_BokehImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_AltairImportHook.find_spec() not found; falling back to find_module()
```



```
#verifico datos nulos
df.isnull().sum()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

```
fecha                   0
dia                     0
hora                    0
latitud                 1
longitud                1
tipologia               0
direccion               0
zona                    0
parroquia_urbana        0
causas                  0
gravedad                0
nro_heridos             0
nro_fallecidos          0
vehiculos_retenidos     0
senalizacion_existente  0
condicion_calzada       0
condicion_atmosferica   0
dtype: int64
```

```
df=df.loc[:,df.columns!="fecha"]
df=df.loc[:,df.columns!="zona"]
df=df.loc[:,df.columns!="latitud"]
df=df.loc[:,df.columns!="longitud"]
df=df.loc[:,df.columns!="direccion"]
df=df.loc[:,df.columns!="gravedad"]
df=df.loc[:,df.columns!="vehiculos_retenidos"]
df=df.loc[:,df.columns!="senalizacion_existente"]
df=df.loc[:,df.columns!="condicion_calzada"]
df=df.loc[:,df.columns!="condicion_atmosferica"]
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

```
df.isnull().sum()
```

```
dia                 0
hora                0
tipologia           0
parroquia_urbana    0
causas              0
nro_heridos         0
nro_fallecidos      0
dtype: int64
```

```
df.sample(10)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos
299	miercoles	h04	estrellamiento	punzara	conducir en exceso de velocidad	0	0
212	sabado	h21	perdida de carril	carigan	conducir en estado de embriaguez	0	0
132	viernes	h18	perdida de carril	sucre	conducir en exceso de velocidad	0	0
293	sabado	h06	estrellamiento	punzara	imprudencia del conductor	1	0
121	martes	h14	atropello	sucre	imprudencia del peaton	1	0
11	viernes	h17	choque lateral angular	sucre	condiciones climaticas desfavorables	0	0
202	domingo	h07	arrollamiento	sucre	imprudencia del peaton	1	0

```
df.head()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos
0	viernes	h11	estrellamiento	el valle	conducir en estado de embriaguez	1	0
1	martes	h17	choque lateral perpendicular	punzara	conducir en estado de embriaguez	1	0
2	martes	h12	choque por alcance	sucre	no mantener la distancia reglamentaria	0	0
3	viernes	h10	atropello	sucre	imprudencia del peaton	1	0

```
ds=pd.DataFrame(df)
#Presenta el numero de filas
print("El numero de filas(observaciones) es: ",ds.shape[0])

#Presenta el numero de columnas
print("El numero de columnas(variables) es: ",len(ds.columns))
```

```
El numero de filas(observaciones) es:  370
El numero de columnas(variables) es:  7
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

```
# tipos de la variables
ds.dtypes
```

```
dia                object
hora               object
tipologia          object
parroquia_urbana   object
causas             object
nro_heridos         int64
nro_fallecidos      int64
dtype: object
```

```
df.shape[0]
```

```
370
```

```
df['causas'].value_counts()
```

```
imprudencia del conductor                161
conducir en estado de embriaguez          88
conducir en exceso de velocidad           75
imprudencia del peaton                    15
no respetar las senales de transito       15
fallas mecanicas no previsibles            6
no mantener la distancia reglamentaria     2
condiciones climaticas desfavorables       2
no ceder el derecho de via                 2
impericia del conductor                    2
cruce de animales en la via                2
Name: causas, dtype: int64
```

```
print(df['causas'].unique())# datos en texto
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer

['conducir en estado de embriaguez'
 'no mantener la distancia reglamentaria' 'imprudencia del peaton'
 'conducir en exceso de velocidad' 'no respetar las senales de transito'
 'condiciones climaticas desfavorables' 'no ceder el derecho de via'
 'impericia del conductor' 'imprudencia del conductor'
 'fallas mecanicas no previsibles' 'cruce de animales en la via']
```

```
print(df['causas'].unique())# datos en texto
```

```
['conducir en estado de embriaguez'
 'no mantener la distancia reglamentaria' 'imprudencia del peaton'
```

```
       'conducir en exceso de velocidad' 'no respetar las senales de transito'
       'condiciones climaticas desfavorables' 'no ceder el derecho de via'
       'impericia del conductor' 'imprudencia del conductor'
       'fallas mecanicas no previsibles' 'cruce de animales en la via']
```

Transformamos los datos a numéricos

```
df['causas']= df['causas'].apply(lambda x:
                                 1 if x == 'conducir en estado de embriaguez' else
                                 2 if x == 'imprudencia del conductor' else
                                 3 if x == 'no ceder el derecho de via' else
                                 4 if x == 'conducir en exceso de velocidad' else
                                 5 if x == 'cambio brusco e indebido de carril' else
                                 6 if x == 'fallas mecanicas no previsibles' else
                                 7 if x == 'no respetar las senales de transito' else
                                 8 if x == 'invadir carril de circulacion' else
                                 9 if x == 'imprudencia del peaton' else
                                 10 if x == 'no mantener la distancia reglamentaria' else
                                 11 if x == 'razones desconocidas' else
                                 12 if x == 'condiciones climaticas desfavorables' else
                                 13 if x == 'negligencia del conductor' else
                                 14 if x == 'no respetar las ordenes del agente de transito' else
                                 15 if x == 'impericia del conductor' else
                                 16)
```

```
print(df['parroquia_urbana'].unique()) # datos convertidos a numeros enteros
```

```
     ['el valle' 'punzara' 'sucre' 'san sebastian' 'carigan' 'el sagrario']
```

```
df['parroquia_urbana']= df['parroquia_urbana'].apply(lambda x:
                                 1 if x == 'el sagrario' else
                                 2 if x == 'san sebastian' else
                                 3 if x == 'el valle' else
                                 4 if x == 'sucre' else
                                 5 if x == 'punzara' else
                                 6 if x == 'carigan' else
                                 7)
```

```
print(df['parroquia_urbana'].unique()) # datos convertidos a numeros enteros
```

```
     [3 5 4 2 6 1]
```

```
print(df['tipologia'].unique()) # datos convertidos a numeros enteros
```

```
     ['estrellamiento' 'choque lateral perpendicular' 'choque por alcance'
      'atropello' 'choque lateral angular' 'roce negativo' 'perdida de carril'
      'perdida de pista' 'choque frontal excentrico' 'colision' 'atipico'
      'arrollamiento' 'volcamiento' 'roce positivo' 'rozamiento'
      'caida de pasajero']
```

```
df['tipologia']= df['tipologia'].apply(lambda x:
                                 1 if x == 'arrollamiento' else
                                 2 if x == 'atipico' else
                                 3 if x == 'atropello' else
                                 4 if x == 'caida de pasajero' else
                                 5 if x == 'choque frontal' else
                                 5 if x == 'choque frontal excentrico' else
                                 5 if x == 'choque frontal longitudinal' else
                                 5 if x == 'choque lateral angular' else
                                 5 if x == 'choque lateral perpendicular' else
                                 5 if x == 'choque por alcance' else
                                 6 if x == 'colision' else
                                 7 if x == 'encunetamiento' else
                                 8 if x == 'estrellamiento' else
                                 9 if x == 'perdida de carril' else
                                 9 if x == 'perdida de pista' else
                                 10 if x == 'roce negativo' else
                                 10 if x == 'roce positivo' else
                                 10 if x == 'rozamiento' else
                                 11 if x == 'volcamiento' else
                                 11 if x == 'volcamiento lateral' else
                                 11)
```

```
print(df['tipologia'].unique()) # datos convertidos a numeros enteros
```

```
     [ 8  5  3 10  9  6  2  1 11  4]
```

```
df['dia']= df['dia'].apply(lambda x:
                                    1 if x == 'lunes' else
                                    1 if x == 'martes' else
                                    1 if x == 'miercoles' else
                                    2 if x == 'jueves' else
                                    2 if x == 'viernes' else
                                    3 if x == 'sabado' else
                                    3 )


print(df['dia'].unique()) # datos convertidos a numeros enteros
```

```
[2 1 3]
```

```
df= df[df['parroquia_urbana'] <= 6]


df['hora']= df['hora'].apply(lambda x:
                                    0 if x == 'h00' else
                                    0 if x == 'h01' else
                                    0 if x == 'h02' else
                                    0 if x == 'h03' else
                                    0 if x == 'h04' else
                                    0 if x == 'h05' else
                                    1 if x == 'h06' else
                                    1 if x == 'h07' else
                                    1 if x == 'h08' else
                                    1 if x == 'h09' else
                                    1 if x == 'h10' else
                                    1 if x == 'h11' else
                                    2 if x == 'h12' else
                                    2 if x == 'h13' else
                                    2 if x == 'h14' else
                                    2 if x == 'h15' else
                                    2 if x == 'h16' else
                                    2 if x == 'h17' else
                                    3 if x == 'h18' else
                                    3 if x == 'h19' else
                                    3 if x == 'h20' else
                                    3 if x == 'h21' else
                                    3 if x == 'h22' else
                                    3)


df.head()
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos
0	2	1	8	3	1	1	0
1	1	2	5	5	1	1	0
2	1	2	5	4	10	0	0
3	2	1	3	4	9	1	0
4	3	1	5	4	1	0	0

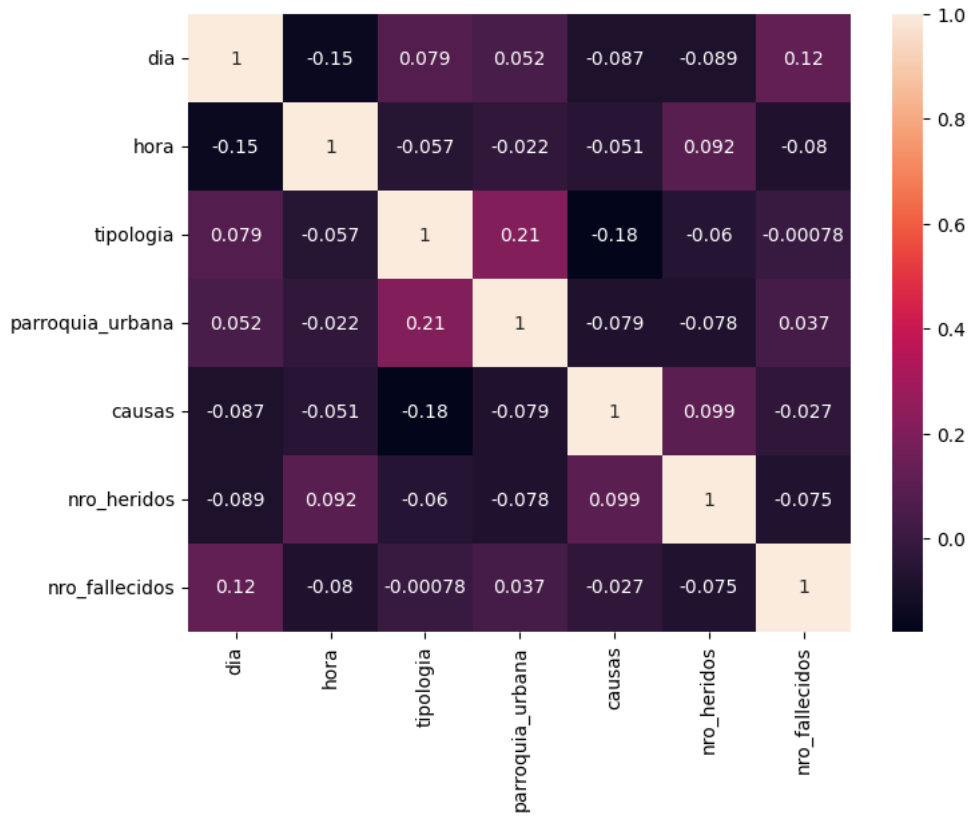
```
# datos aleatorios (muestra de 10 elementos)

df.sample(10)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t
```

Verificacion de correlacion de variables

```
#corelacion
corr_df = df.corr(method='pearson')

plt.figure(figsize=(8, 6))
sns.heatmap(corr_df, annot=True)
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t

<frozen importlib._bootstrap>:914: ImportWarning:

APICoreClientInfoImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_PyDriveImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_OpenCVImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_BokehImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_AltairImportHook.find_spec() not found; falling back to find_module()
```



Seleccionamos la columna que vamos a predecir los datos

```
X = df.iloc[:, [0,1,3,4,5,6]] # atributos de entrada seran las primeras columnas
Y = df.iloc[:,[2]] # atributos de destino

#presentacion de los atributos de entrada
X.head()
```

	dia	hora	parroquia_urbana	causas	nro_heridos	nro_fallecidos
0	2	1	3	1	1	0
1	1	2	5	1	1	0
2	1	2	4	10	0	0
3	2	1	4	9	1	0

```
feature_names = X.columns.tolist()

# Imprimir los nombres de las características
print(feature_names)
```

```
['dia', 'hora', 'parroquia_urbana', 'causas', 'nro_heridos', 'nro_fallecidos']
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

Revisamos la columna a predecir seleccionada

```
#presentacion de los atributos de destino
Y.head()
```

	tipologia
0	8
1	5
2	5
3	3
4	5

Cargamos de modelo 2018-2020

```
from joblib import load
arbol_modelo = load('modelo_tipologia.joblib')
y_pred = arbol_modelo.predict(X)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

Verificamos los datos a predecir

```
print(y_pred)
```

```
[5 5 5 3 5 5 5 8 5 5 5 5 5 5 5 5 8 5 5 5 5 5 5 8 5 8 5 5 5 5 5 5 5 5 5 5
 8 8 5 5 5 5 5 5 8 5 5 8 5 5 5 5 5 5 5 8 5 5 5 5 5 5 5 5 8 5 8 5 5 5 5 5
 5 5 5 5 5 5 8 8 5 5 5 5 5 8 5 5 5 5 5 5 5 5 8 5 5 5 5 5 5 5 5 5 5 8 5 8
 5 5 5 5 5 5 5 5 5 8 3 5 5 8 8 5 5 5 3 5 5 8 5 8 5 5 5 5 5 5 3 5 5 5 8 8 5
 3 5 5 5 5 5 5 5 5 3 8 5 5 5 5 5 5 5 5 3 8 5 5 5 5 3 5 5 5 5 5 5 5 5 5 8
 5 8 5 5 5 5 8 5 5 5 5 8 5 8 5 5 8 3 5 5 5 5 8 5 5 3 5 5 5 5 5 5 5 3 5 5 5
 5 5 5 5 5 8 5 5 8 8 5 5 8 5 5 5 8 5 5 5 5 5 5 5 3 5 5 5 8 5 8 5 5 5 5 5 5
 5 5 8 5 5 5 8 8 3 5 5 5 5 5 5 8 5 5 8 5 5 5 5 5 8 5 8 5 5 8 5 5 5 5 5 5 5
 5 5 5 8 5 5 5 5 5 5 5 5 8 5 5 5 5 5 5 5 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 8 5 8 5 3 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5]
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

Configuramos la metrica de clasificación

```
# Resumen de las predicciones hechas por el clasificador
from sklearn import metrics
reporte = metrics.classification_report(Y, y_pred,output_dict=True)
pre = pd.DataFrame(reporte).transpose()
print(pre)
pre.to_excel("resumen_precision_tipologia_2021.xlsx")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cc

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cc

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to cc

<frozen importlib._bootstrap>:914: ImportWarning:

APICoreClientInfoImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_PyDriveImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_OpenCVImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_BokehImportHook.find_spec() not found; falling back to find_module()

<frozen importlib._bootstrap>:914: ImportWarning:

_AltairImportHook.find_spec() not found; falling back to find_module()

              precision    recall  f1-score     support
1              0.000000  0.000000  0.000000    3.000000
2              0.000000  0.000000  0.000000   10.000000
3              0.866667  0.650000  0.742857   20.000000
4              0.000000  0.000000  0.000000    1.000000
5              0.552980  0.976608  0.706131  171.000000
6              0.000000  0.000000  0.000000   10.000000
8              0.660377  0.307018  0.419162  114.000000
9              0.000000  0.000000  0.000000   26.000000
10             0.000000  0.000000  0.000000   14.000000
11             0.000000  0.000000  0.000000    1.000000
accuracy       0.581081  0.581081  0.581081    0.581081
macro avg      0.208002  0.193363  0.186815  370.000000
weighted avg   0.505881  0.581081  0.495649  370.000000
```

Predicciones de los datos obtenidas

```
pred = pd.DataFrame(y_pred)
pred.head()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t
```

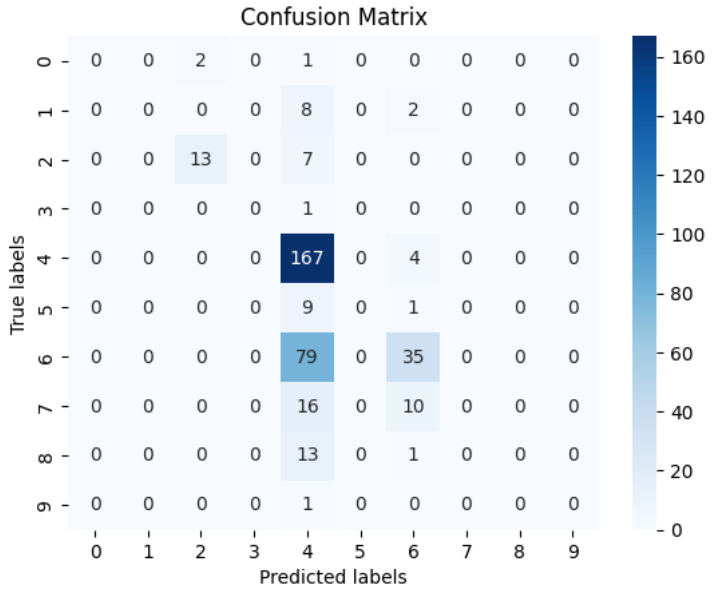
	0
0	5
1	5
2	5
3	3
4	5

Generamos la matriz de confusión

```
#Matriz de confusion
matriz=confusion_matrix(Y, y_pred)
ax= plt.subplot()
sns.heatmap(matriz, annot=True, cmap="Blues",fmt='g');
```

```
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix');
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t
```



Concatenamos los datos de las horas originales con los datos predichos

```
#Para concatenar variables
import pandas as pd

prediccion = pd.DataFrame(y_pred,columns=['tipologia_prediccion'])
original = Y
original.reset_index(drop=True, inplace=True)
df_combined = pd.concat([prediccion,original], axis=1)
df_combined.head(10)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t
```

	tipologia_prediccion	tipologia
0	5	8
1	5	5
2	5	5
3	3	3
4	5	5
5	5	5
6	5	8
7	8	8
8	5	8
9	5	5

Visualizamos la exactitud del modelo

```
# Precisión
from sklearn.metrics import accuracy_score, recall_score, precision_score
print('Exactitud árboles de decisión: ',accuracy_score(pred,Y)*100)
print('Exhaustividad árboles de decisión: ', recall_score(pred,Y,average='micro')*100)
print('Precision árboles de decisión: ',accuracy_score(pred,Y)*100)
```

```
Exactitud árboles de decisión:  58.108108108108105
Exhaustividad árboles de decisión:  58.108108108108105
Precision árboles de decisión:  58.108108108108105
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

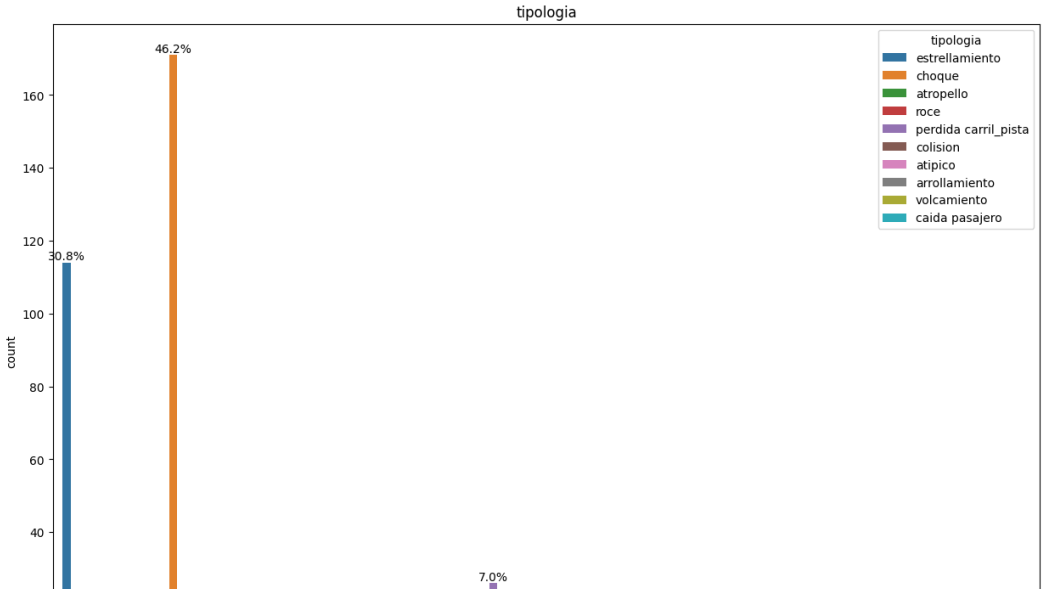
Transformamos las variables

```
#transformar datos para visualización de variables predictoras y originales
df_combined['tipologia']= df_combined['tipologia'].apply(lambda x:
                                    'arrollamiento' if x == 1 else
                                    'atipico' if x == 2 else
                                    'atropello' if x == 3 else
                                    'caida pasajero' if x == 4 else
                                    'choque' if x == 5 else
                                    'colision' if x == 6 else
                                    'encunetamiento' if x == 7 else
                                    'estrellamiento' if x == 8 else
                                    'perdida carril_pista' if x == 9 else
                                    'roce' if x == 10 else
                                    'volcamiento')

df_combined['tipologia_prediccion']= df_combined['tipologia_prediccion'].apply(lambda x:
                                    'arrollamiento' if x == 1 else
                                    'atipico' if x == 2 else
                                    'atropello' if x == 3 else
                                    'caida pasajero' if x == 4 else
                                    'choque' if x == 5 else
                                    'colision' if x == 6 else
                                    'encunetamiento' if x == 7 else
                                    'estrellamiento' if x == 8 else
                                    'perdida carril_pista' if x == 9 else
                                    'roce' if x == 10 else
                                    'volcamiento')
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

Graficamos las estadisticas originales de los datos

```
import seaborn as sns #permite generar fácilmente elegantes gráficos, proporciona una interfaz de alto nivel que es realmente sencilla d
ax = plt.subplots(figsize = (15,10))
ncount=len(df_combined)
#ax[1].plot(df['dia'],df['hora'],color = 'tab:purple')
sns.countplot(x='tipologia',hue = 'tipologia', data = df_combined, ax = ax[1]) #Muestre el conteo de observaciones en cada contenedor ca
ax[1].set_title('tipologia')



for p in ax[1].patches:
    x=p.get_bbox().get_points()[:,0]
    y=p.get_bbox().get_points()[1,1]
    ax[1].annotate('{:.1f}%'.format(100.*y/ncount), (x.mean(), y),
            ha='center', va='bottom') # set the alignment of the text

#Guardamos la figura de barras original
ax[0].savefig("barras_original_tipologia_2021.png")
```



Graficamos las estadisticas predecidas de los datos

```
ax = plt.subplots(figsize = (15,10))
ncount=len(df_combined)
sns.countplot(x='tipologia_prediccion',hue = 'tipologia_prediccion', data = df_combined, ax = ax[1]) #Muestre el conteo de observaciones
ax[1].set_title('tipologia_prediccion')


for p in ax[1].patches:
    x=p.get_bbox().get_points()[:,0]
    y=p.get_bbox().get_points()[1,1]
    ax[1].annotate('{:.1f}%'.format(100.*y/ncount), (x.mean(), y),
            ha='center', va='bottom') # set the alignment of the text

#Guardamos la figura de barras de predicción
ax[0].savefig("barras_prediccion_tipologia_2021.png")
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t
```

tipologia_prediccion

tipologia_prediccion

Crear el arbol de desición actual con el modelo con datos predichos del 2021

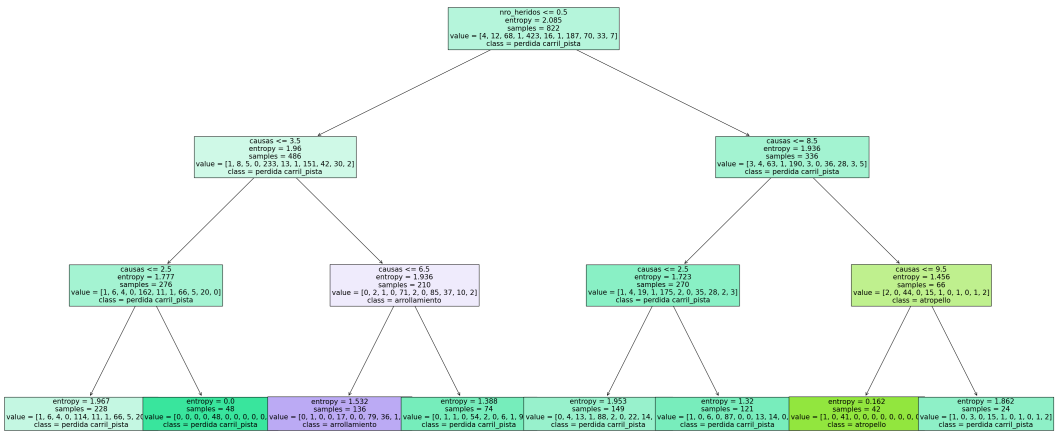
```
#target_names_str = [str(name) for name in target_names]
variable = df['tipologia'].apply(lambda x:
                                 'arrollamiento' if x == 1 else
                                 'atipico' if x == 2 else
                                 'atropello' if x == 3 else
                                 'caida pasajero' if x == 4 else
                                 'choque' if x == 5 else
                                 'colision' if x == 6 else
                                 'encunetamiento' if x == 7 else
                                 'estrellamiento' if x == 8 else
                                 'perdida carril_pista' if x == 9 else
                                 'roce' if x == 10 else
                                 'volcamiento')

claseVar = variable.unique().tolist()
target_names_str = [str(name) for name in claseVar]
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

tipologia_prediccion

Creamos la figura del arbol

```
figura_arbol = plt.figure(figsize=(40,20)) # Le indicamos las dimensiones que queremos que tenga
plot_tree(arbol_modelo,feature_names=feature_names,filled=True, class_names=target_names_str, fontsize=15)
plt.show()
```




```
figura_arbol.savefig("arbol_colab_tipologia_2021.png")
```

Verificamos la probabilidad de los accidentes de tránsito a través de la predicción del modelo

```
#Verificamos la probabilidad de todas las variables
y_proba = arbol_modelo.predict_proba(X)
print(y_proba.shape)
print(y_pred.shape)
probabilidad_acierto = np.round(y_proba[0][y_pred] * 100, 2)
print("Probabilidad de Acierto: " + str(probabilidad_acierto) + "%")
```

```
(370, 11)
(370,)
Probabilidad de Acierto: [1.34 1.34 1.34 0.67 1.34 1.34 1.34 9.4  1.34 1.34 1.34 1.34 1.34 1.34
 1.34 1.34 9.4  1.34 1.34 1.34 1.34 1.34 1.34 9.4  1.34 9.4  1.34 1.34
 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 9.4  9.4  1.34 1.34 1.34
 1.34 1.34 1.34 9.4  1.34 1.34 9.4  1.34 1.34 1.34 1.34 1.34 1.34 1.34
 9.4  1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 9.4  1.34 9.4  1.34
 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 9.4  9.4  1.34 1.34
 1.34 1.34 1.34 9.4  1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 9.4  1.34
 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 9.4  1.34 9.4  1.34
 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 9.4  0.67 1.34 1.34 9.4  9.4
 1.34 1.34 1.34 0.67 1.34 1.34 9.4  1.34 9.4  1.34 1.34 1.34 1.34 1.34
 1.34 0.67 1.34 1.34 1.34 9.4  9.4  1.34 0.67 1.34 1.34 1.34 1.34 1.34
 1.34 1.34 1.34 0.67 9.4  1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 0.67
 9.4  1.34 1.34 1.34 1.34 0.67 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34
 1.34 1.34 9.4  1.34 9.4  1.34 1.34 1.34 1.34 9.4  1.34 1.34 1.34 1.34
 9.4  1.34 9.4  1.34 1.34 9.4  0.67 1.34 1.34 1.34 1.34 9.4  1.34 1.34
 0.67 1.34 1.34 1.34 1.34 1.34 1.34 1.34 0.67 1.34 1.34 1.34 1.34 1.34
 1.34 1.34 1.34 9.4  1.34 1.34 9.4  9.4  9.4  1.34 1.34 9.4  1.34 1.34
 1.34 9.4  1.34 1.34 1.34 1.34 1.34 1.34 1.34 0.67 1.34 1.34 1.34 9.4  1.34
 9.4  1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 9.4  1.34 1.34 1.34 9.4
 9.4  0.67 1.34 1.34 1.34 1.34 1.34 1.34 9.4  1.34 1.34 9.4  1.34 1.34
 1.34 1.34 1.34 9.4  1.34 9.4  1.34 1.34 9.4  1.34 1.34 1.34 1.34 1.34
 1.34 1.34 1.34 1.34 1.34 9.4  1.34 1.34 1.34 1.34 1.34 1.34 1.34 9.4
 1.34 1.34 1.34 1.34 1.34 1.34 1.34 0.67 1.34 1.34 1.34 1.34 1.34 1.34
 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34
 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 9.4  1.34
 9.4  1.34 0.67 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34 1.34
 1.34 1.34 1.34 1.34 1.34 1.34]%
```

```
print(type(probabilidad_acierto))
columna_probabilidades = pd.DataFrame()
columna_probabilidades['probabilidad'] = pd.DataFrame(probabilidad_acierto)
print(columna_probabilidades)
```

```
<class 'numpy.ndarray'>
     probabilidad
0            1.34
1            1.34
2            1.34
3            0.67
4            1.34
..            ...
365          1.34
366          1.34
367          1.34
368          1.34
369          1.34

[370 rows x 1 columns]
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argumer
```

Presentamos la probabilidad de los datos

```
y_prediccion = pd.DataFrame()
y_prediccion['predicciones'] = pd.DataFrame(y_pred)

probabilidades_2021_df = pd.concat([df,columna_probabilidades, y_prediccion], axis=1)
probabilidades_2021_df.sample(10)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t
```

```
probabilidades_2021_df['predicciones']= probabilidades_2021_df['predicciones'].apply(lambda x:
                                    'arrollamiento' if x == 1 else
                                    'atipico' if x == 2 else
                                    'atropello' if x == 3 else
                                    'caida pasajero' if x == 4 else
                                    'choque' if x == 5 else
                                    'colision' if x == 6 else
                                    'encunetamiento' if x == 7 else
                                    'estrellamiento' if x == 8 else
                                    'perdida carril_pista' if x == 9 else
                                    'roce' if x == 10 else
                                    'volcamiento')

probabilidades_2021_df['dia']= probabilidades_2021_df['dia'].apply(lambda x:
                                    'lunes_marte_miercoles' if x == 1 else
                                    'jueves_viernes' if x == 2 else
                                    'sabado_domingo')

probabilidades_2021_df['hora']= probabilidades_2021_df['hora'].apply(lambda x:
                                    'h00-h05' if x == 0 else
                                    'h06-h011' if x == 1 else
                                    'h012-h17' if x == 2 else
                                    'h18-h23')

probabilidades_2021_df['parroquia_urbana']= probabilidades_2021_df['parroquia_urbana'].apply(lambda x:
                                    'el sagrario' if x == 1 else
                                    'san sebastian' if x == 2 else
                                    'el valle' if x == 3 else
                                    'sucre' if x == 4 else
                                    'punzara' if x == 5 else
                                    'carigan')

probabilidades_2021_df['tipologia']= probabilidades_2021_df['tipologia'].apply(lambda x:
                                    'arrollamiento' if x == 1 else
                                    'atipico' if x == 2 else
                                    'atropello' if x == 3 else
                                    'caida pasajero' if x == 4 else
                                    'choque' if x == 5 else
                                    'colision' if x == 6 else
                                    'encunetamiento' if x == 7 else
                                    'estrellamiento' if x == 8 else
                                    'perdida carril_pista' if x == 9 else
                                    'roce' if x == 10 else
                                    'volcamiento')


probabilidades_2021_df['causas']= probabilidades_2021_df['causas'].apply(lambda x:
                                    'conducir en estado de embriaguez' if x == 1 else
                                    'imprudencia del conductor' if x == 2 else
                                    'no ceder el derecho de via' if x == 3 else
                                    'conducir en exceso de velocidad' if x == 4 else
                                    'cambio brusco e indebido de carril' if x == 5 else
                                    'fallas mecanicas no previsibles' if x == 6 else
                                    'no respetar las senales de transito' if x == 7 else
                                    'invadir carril de circulacion' if x == 8 else
                                    'imprudencia del peaton' if x == 9 else
                                    'no mantener la distancia reglamentaria' if x == 10 else
                                    'razones desconocidas' if x == 11 else
                                    'condiciones climaticas desfavorables' if x == 12 else
                                    'negligencia del conductor' if x == 13 else
                                    'no respetar las ordenes del agente de transito' if x == 14 else
                                    'impericia del conductor' if x == 15 else
                                    'cruce de animales en la via')

probabilidades_2021_df.sample(10)
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:

`should_run_async` will not call `transform_cell` automatically in the future. Please pass the result t
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecido
356	lunes_marte_miercoles	h18-h23	choque	sucre	imprudencia del conductor	3	
97	sabado_domingo	h06-h011	choque	el sagrario	imprudencia del conductor	0	
274	sabado_domingo	h00-h05	estrellamiento	el sagrario	conducir en exceso de velocidad	0	
115	jueves_viernes	h06-h011	colision	punzara	conducir en estado de embriaguez	0	
90	lunes_marte_miercoles	h012-h17	choque	punzara	conducir en exceso de velocidad	1	
329	sabado_domingo	h00-h05	estrellamiento	el sagrario	imprudencia del conductor	0	

```
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages


# Crear una figura y un eje
fig, ax = plt.subplots(figsize=(8, 6))

# Eliminar marcas del eje
ax.axis('off')

# Crear la tabla a partir del DataFrame
tabla = pd.plotting.table(ax, probabilidades_2021_df, loc='center', cellLoc='center', fontsize=14)

# Guardar la tabla en un archivo PDF
with PdfPages('tabla_probabilidades_tipologia_2021.pdf') as pdf:
    pdf.savefig(fig, bbox_inches='tight')
```