

Variable "causas"

Cargamos las librerías necesarias para la elaboración 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; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: APICoreClientInfoImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _PyDriveImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _OpenCVImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _BokehImportHook.find_spec() not found; falling back to user site directory
<frozen importlib._bootstrap>:914: ImportWarning: _AltairImportHook.find_spec() not found; falling back to user site directory
```

	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 estadi embriac
1	01-05	martes	h17	-4.020370	-79.217962	choque lateral perpendicular	benjamin carrion y gustavo serrano	urbana	punzara	conduci estadi embriac
2	01-05	martes	h12	-3.987230	-79.202984	choque por alcance	nueva loja y guaranda	urbana	sucres	no mante la dista reglamenti
3	01-08	viernes	h10	-3.989410	-79.236506	atropello	angel felicisimo rojas	urbana	sucres	imprude del pei
4	01-09	sabado	h06	-3.979784	-79.218689	choque lateral angular	isidro ayora y habana	urbana	sucres	conduci estadi embriac



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 to `transformed_cell` argumer
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos
257	lunes	h21	perdida de carril	el valle	conducir en exceso de velocidad	1	0
144	sabado	h15	atropello	sucre	imprudencia del conductor	1	0
38	domingo	h04	estrellamiento	sucre	conducir en exceso de velocidad	0	0
115	viernes	h08	colision	punzara	conducir en estado de embriaguez	0	0
97	domingo	h11	choque lateral perpendicular	el sagrario	imprudencia del conductor	0	0
283	miercoles	h04	atipico	punzara	fallas mecanicas no previsibles	0	0
289	viernes	h10	choque lateral angular	sucre	imprudencia del conductor	0	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 to
```

	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
```

```
< [Progress bar]
```

```
# 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
```

```
['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']
```

```
/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
```

```
< [Progress bar]
```

```
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 )
```

```
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
    6 if x == 'choque frontal excentrico' else
    7 if x == 'choque frontal longitudinal' else
    8 if x == 'choque lateral angular' else
    9 if x == 'choque lateral perpendicular' else
    10 if x == 'choque por alcance' else
    11 if x == 'colision' else
    12 if x == 'encunetamiento' else
    13 if x == 'estrellamiento' else
    14 if x == 'perdida de carril' else
    15 if x == 'perdida de pista' else
    16 if x == 'roce negativo' else
    17 if x == 'roce positivo' else
    18 if x == 'rozamiento' else
    19 if x == 'volcamiento' else
    20 if x == 'volcamiento lateral' else
    21)
```

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

```
[13 9 10 3 8 16 14 15 6 11 2 1 19 17 18 4]
```

```
df['dia']= df['dia'].apply(lambda x:
    1 if x == 'lunes' else
    2 if x == 'martes' else
    3 if x == 'miercoles' else
    4 if x == 'jueves' else
    5 if x == 'viernes' else
    6 if x == 'sabado' else
    7 )
```

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

```
[5 2 6 7 1 4 3]
```

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

```
df['hora']= df['hora'].apply(lambda x:
    0 if x == 'h00' else
    1 if x == 'h01' else
    2 if x == 'h02' else
    3 if x == 'h03' else
    4 if x == 'h04' else
    5 if x == 'h05' else
    6 if x == 'h06' else
    7 if x == 'h07' else
    8 if x == 'h08' else
    9 if x == 'h09' else
    10 if x == 'h10' else
    11 if x == 'h11' else
    12 if x == 'h12' else
    13 if x == 'h13' else
    14 if x == 'h14' else
    15 if x == 'h15' else
    16 if x == 'h16' else
    17 if x == 'h17' else
    18 if x == 'h18' else
    19 if x == 'h19' else
    20 if x == 'h20' else
    21 if x == 'h21' else
    22 if x == 'h22' else
    23)
```

```
df.head()
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos
0	5	11	13	3	1	1	0
1	2	17	9	5	1	1	0
2	2	12	10	4	10	0	0
3	5	10	3	4	9	1	0
4	6	6	8	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 1
```

Verificamos la correlacion de los datos

```
100 7 10 12 6 2 1 0
#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 1
```

```
<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:
```

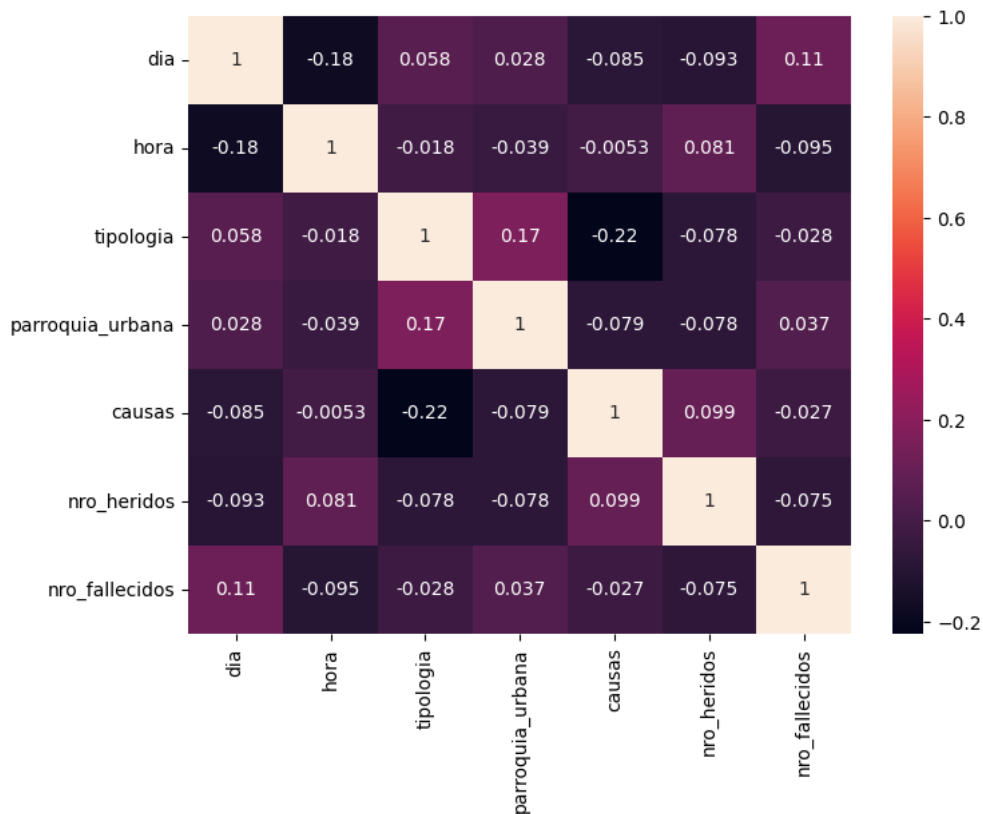
```
_OpenCVImportHook.find_spec() not found; falling back to find_module()
```

```
<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()
```



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

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

	dia	hora	tipologia	parroquia_urbana	nro_heridos	nro_fallecidos
0	5	11	13	3	1	0
1	2	17	9	5	1	0

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

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

```
['dia', 'hora', 'tipologia', 'parroquia_urbana', '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
```

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

	causas
0	1
1	1
2	10
3	9
4	1

Cargamos el modelo

```
from joblib import load
arbol_modelo = load('modelo_causas.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
```

```
print(y_pred)
```

```
[ 4  3 10  9  3 10  4  4  4  3  4  3  3  3  2  3  4  3  4  3  3 10  4  4
  3  3  3  3  3  4  3  3  3  3  4  4  1  4  4  4  3  4  4  4  4  4  3  3
  4  2  4  3  3  4  4 10  4  3  3  3  3  9  3  3  4  3  4  3  2  3  3  3
  2  4  2  3  4  3  1  3  3  4  4  4  3 10  2  3  4  4  3  4  1  4  4  3
  4  3  4  3  4  3  4  4  3  2 10  3  3 10  4  3  2  9  4  2  2  3  3  3
  4  9  4  4  4  3  4  3  4  3  9  4 10  4 10  4  4 10  3  3  2  4  9  4 10
  9  4  4  4  9  1  2  2  3  3  3  2  4  9  4  4  3  3  3  3  2  3  3  9
  4  4  1  3  2  9  3  9  3  2  4  4  3  3  2 10  4  4  4  4  4  2  3  4
  4  3  3  4  4  3  4  4  3  4  9  9  3  3  2  4 10  4  9  3  4 10  3  3
  3  4  9  3  3  4  4  3  3  4  3  4 10  3  4  4  4  4  3  4  2  2  9  4
  3  3  3  4  3 10  9  3  2  3  4  2  2  3  4  4  3  4 10  3  2  4  9  3
 10  2  4  9  4  3  3  2  1  2  4  4  3  4  3  3  4  3  3  2  3  4  4  3
  4  3  3  4  4  4  3  9  3  4  4  4  4  4  2  4  4  4  4  4  3  4  3
 10  1 10  9  3  3  4  3  4  3  4  4  4  4  4  4  4  4  4  3  3  4  3 10
  4  4  3  4  3  4  3  3  3  3  9  4  3  4  3  9  3  2 10  3 10  2  2
  2 10  3  3  3  3 10  3  3 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 to `transformed_cell` argumer
```

```
# 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_causas_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 co
```

```
/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         0.142857   0.011364   0.021053     88.000000
2         0.571429   0.124224   0.204082    161.000000
3         0.014493   1.000000   0.028571      2.000000
4         0.407143   0.760000   0.530233    75.000000
6         0.000000   0.000000   0.000000      6.000000
7         0.000000   0.000000   0.000000    15.000000
9         0.625000   1.000000   0.769231    15.000000
10        0.076923   1.000000   0.142857      2.000000
12        0.000000   0.000000   0.000000      2.000000
15        0.000000   0.000000   0.000000      2.000000
16        0.000000   0.000000   0.000000      2.000000
accuracy   0.262162   0.262162   0.262162      0.262162
macro avg   0.167077   0.354144   0.154184   370.000000
weighted avg 0.390986   0.262162   0.233401   370.000000
```

Pedicciones

```
#Predicciones
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 i
```

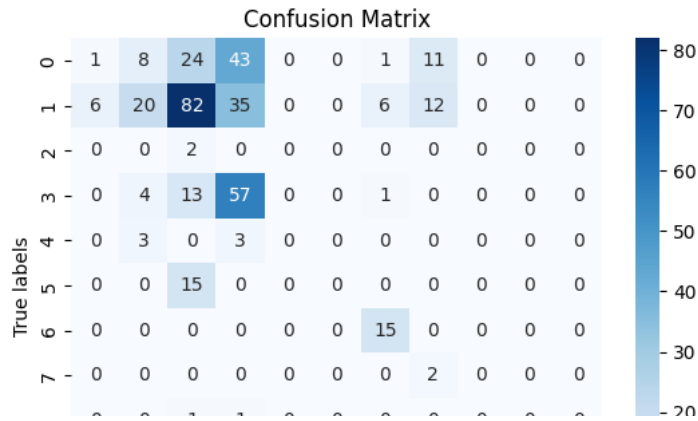
	0	1	2
0	4		
1	3		
2	10		
3	9		
4	3		

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 1
```



```
#Para concatenar variables
```

```
import pandas as pd
```

```
prediccion = pd.DataFrame(y_pred,columns=['causas_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 1
```

	causas_prediccion	causas
0	4	1
1	3	1
2	10	10
3	9	9
4	3	1
5	10	10
6	4	4
7	4	4
8	4	4
9	3	7

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('Precisión árboles de decisión: ',accuracy_score(pred,Y)*100)
```

```
Exactitud árboles de decisión: 26.216216216216214
```

```
Exhaustividad árboles de decisión: 26.216216216216214
```

```
Precisión árboles de decisión: 26.216216216216214
```

```
/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['causas_prediccion']= df_combined['causas_prediccion'].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')

df_combined['causas']= df_combined['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')

/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 estadísticas originales de los datos

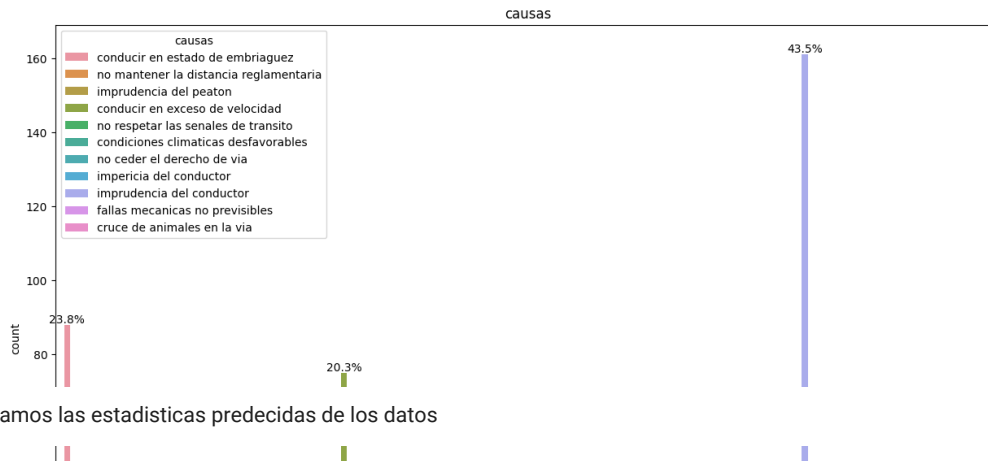
```

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)
sns.countplot(x='causas',hue = 'causas', data = df_combined, ax = ax[1]) #Muestre el conteo de observaciones en cada contenedor categóri
ax[1].set_title('causas')

for p in ax[1].patches:
    x=p.get_bbox().get_points()[0,0]
    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_causas_2021.png")

```



Graficamos las estadísticas predecidas de los datos

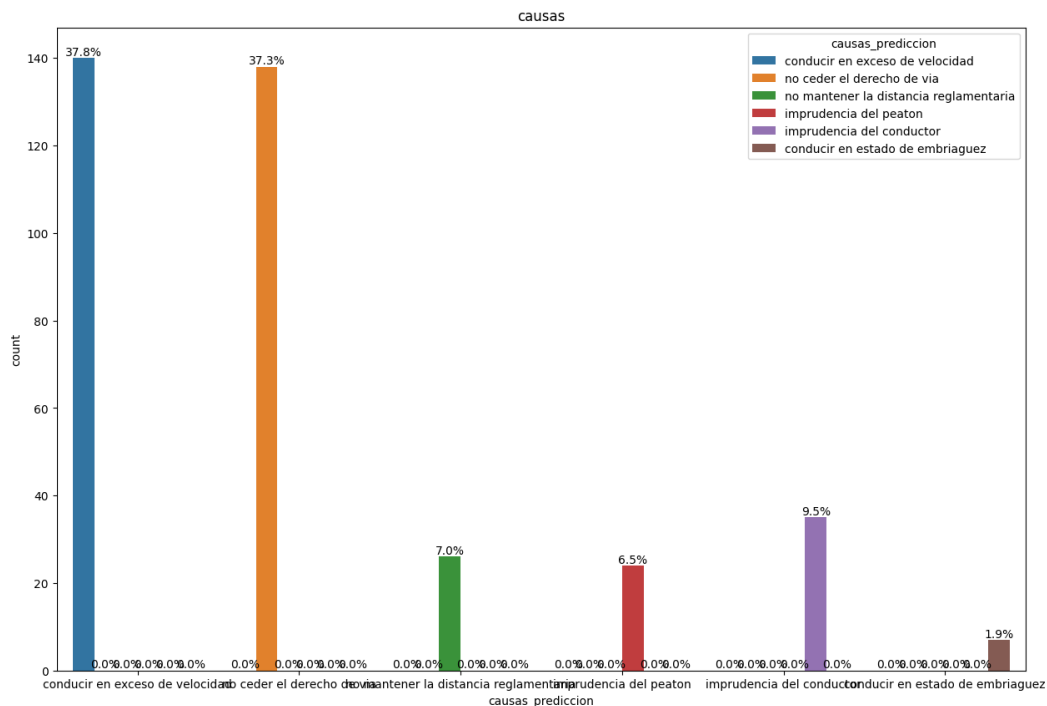
```
ax = plt.subplots(figsize = (15,10))
ncount=len(df_combined)
sns.countplot(x='causas_prediccion',hue = 'causas_prediccion', data = df_combined, ax = ax[1]) #Muestre el conteo de observaciones en ca
ax[1].set_title('causas')
```

```
for p in ax[1].patches:
    x=p.get_bbox().get_points()[0,0]
    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_causas_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 f



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

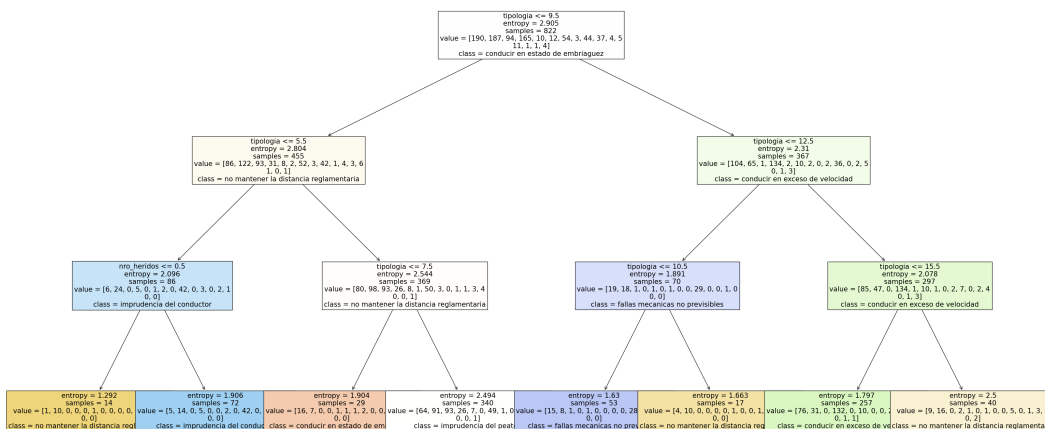
```
#target_names_str = [str(name) for name in target_names]
variable= df['causas'].apply(lambda x:
    'conducir en estado de embriaguez' if x == 1 else
    'imprudencia del conductor' if x == 2 else
    'no ceder el derecho de via' if x == 3 else
    'conducir en exceso de velocidad' if x == 4 else
    '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')

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
```



Creamos la figura del arbol

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



```
figura_arbol.savefig("arbol_colab_causas_2021.png")
```

```
X.head()
```

dia hora tipologia parroquia_urbana nro_heridos nro_fallecidos  

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

1 2 17 9 5 1 0

#Verificamos la probabilidad de todas las variables

y_proba = arbol_modelo.predict_proba(X)

probabilidad_acierto = np.round(y_proba[0][y_pred] * 100, 2)

print("Probabilidad de Acierto: " + str(probabilidad_acierto) + "%")

```
Probabilidad de Acierto: [ 0.  51.36 0.  0.78 51.36 0.  0.  0.  0.  51.36 0.  51.36
51.36 51.36 0.  51.36 0.  51.36 0.  51.36 51.36 0.  0.  0.
51.36 51.36 51.36 51.36 51.36 0.  51.36 51.36 51.36 51.36 0.  0.
12.06 0.  0.  0.  51.36 0.  0.  0.  0.  0.  51.36 51.36
0.  0.  0.  51.36 51.36 0.  0.  0.  0.  51.36 51.36 51.36
51.36 0.78 51.36 51.36 0.  51.36 0.  51.36 0.  51.36 51.36 51.36
0.  0.  0.  51.36 0.  51.36 12.06 51.36 51.36 0.  0.  0.
51.36 0.  0.  51.36 0.  0.  51.36 0.  12.06 0.  0.  51.36
0.  51.36 0.  51.36 0.  51.36 0.  51.36 0.  0.  51.36
51.36 0.  0.  51.36 0.  0.78 0.  0.  0.  51.36 51.36 51.36
0.  0.78 0.  0.  0.  0.  51.36 0.  51.36 0.78 0.  0.
0.  0.  0.  0.  0.  51.36 51.36 0.  0.  0.78 0.  0.
0.78 0.  0.  0.78 12.06 0.  0.  51.36 51.36 51.36 0.
0.  0.78 0.  0.  51.36 51.36 51.36 51.36 0.  51.36 51.36 0.78
0.  0.  12.06 51.36 0.  0.78 51.36 0.78 51.36 0.  0.  0.
51.36 51.36 0.  0.  0.  0.  0.  0.  0.  0.  51.36 0.
0.  51.36 51.36 51.36 0.  0.  0.  51.36 0.  0.  0.78 0.78
0.  51.36 0.  0.  0.  0.  0.78 51.36 0.  0.  51.36 51.36
51.36 0.  0.78 51.36 51.36 0.  0.  51.36 51.36 0.  51.36 0.
0.  51.36 0.  0.  51.36 0.  0.  51.36 0.  0.  0.78 0.
51.36 51.36 51.36 0.  51.36 0.  0.78 51.36 0.  51.36 0.  0.
0.  51.36 0.  0.  51.36 0.  0.  51.36 0.  0.  0.78 51.36
0.  0.  0.  0.78 0.  51.36 51.36 0.  12.06 0.  0.  0.
51.36 0.  51.36 51.36 0.  51.36 51.36 0.  51.36 0.  0.  51.36
0.  51.36 51.36 0.  0.  0.  51.36 0.78 51.36 0.  0.  0.
0.  0.  0.  0.  0.  0.  0.  0.  51.36 0.  51.36 0.
0.  0.  51.36 0.  51.36 0.  51.36 51.36 51.36 51.36 51.36 0.78
0.  51.36 0.  51.36 0.78 51.36 0.  0.  51.36 0.  0.  0.
0.  0.  51.36 51.36 51.36 0.  51.36 51.36 0.  51.36 0.  0. ]%
```

/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

◀ ▶

print(type(probabilidad_acierto))

columna_probabilidades = pd.DataFrame()

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

print(columna_probabilidades)

```
<class 'numpy.ndarray'>
probabilidad
0      0.00
1     51.36
2      0.00
3     0.78
4     51.36
..      ...
365    51.36
366    0.00
367    51.36
368    51.36
369    0.00
```

[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 of
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos	probabilidad	predicc
135	6	15	13		5	4	1	0	0.00
237	1	13	18		4	1	0	0	0.00
202	7	7	1		4	9	1	0	0.78
243	5	19	13		4	1	0	0	0.00
161	3	9	9		2	2	0	0	51.36
41	3	10	14		3	12	1	0	0.00
302	2	20	11		6	2	2	0	0.00
60	5	15	9		1	2	0	0	51.36
66	5	23	13		4	4	0	0	0.00

```
probabilidades_2021_df['predicciones']= probabilidades_2021_df['predicciones'].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['dia']= probabilidades_2021_df['dia'].apply(lambda x:
    'lunes' if x == 1 else
    'martes' if x == 2 else
    'miercoles' if x == 3 else
    'jueves' if x == 4 else
    'viernes' if x == 5 else
    'sabado' if x == 6 else
    'domingo' )
```

```
probabilidades_2021_df['hora']= probabilidades_2021_df['hora'].apply(lambda x:
    'h00' if x == 0 else
    'h01' if x == 1 else
    'h02' if x == 2 else
    'h03' if x == 3 else
    'h04' if x == 4 else
    'h05' if x == 5 else
    'h06' if x == 6 else
    'h07' if x == 7 else
    'h08' if x == 8 else
    'h09' if x == 9 else
    'h10' if x == 10 else
    'h11' if x == 11 else
    'h12' if x == 12 else
    'h13' if x == 13 else
    'h14' if x == 14 else
    'h15' if x == 15 else
    'h16' if x == 16 else
    'h17' if x == 17 else
    'h18' if x == 18 else
    'h19' if x == 19 else
    'h20' if x == 20 else
    'h21' if x == 21 else
    'h22' if x == 22 else
    '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 frontal' if x == 5 else
    'choque frontal excentrico' if x == 6 else
    'choque frontal longitudinal' if x == 7 else
    'choque lateral angular' if x == 8 else
    'choque lateral perpendicular' if x == 9 else
    'choque alcance' if x == 10 else
    'colision' if x == 11 else
    'encunetamiento' if x == 12 else
    'estrellamiento' if x == 13 else
    'perdida carril' if x == 14 else
    'perdida pista' if x == 15 else
    'roce negativo' if x == 16 else
    'roce posositivo' if x == 17 else
    'rozamiento' if x == 18 else
    'volcamiento' if x == 19 else
    'volcamiento lateral' if x == 20 else
    'volcamiento longitudinal')

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_fallecidos	probabilidad
162	miercoles	h17	choque lateral angular	el sagrario	conducir en estado de embriaguez	0	0	51.
61	miercoles	h12	atropello	el sagrario	imprudencia del conductor	1	0	0.
101	martes	h21	choque lateral angular	el sagrario	conducir en exceso de velocidad	1	0	51.
45	domingo	h02	estrellamiento	el sagrario	conducir en exceso de velocidad	0	0	0.
319	martes	h14	choque lateral perpendicular	el sagrario	imprudencia del conductor	1	0	51.
67	lunes	h05	choque lateral perpendicular	san sebastian	imprudencia del conductor	2	0	51.
143	viernes	h23	choque alcance	sucre	conducir en estado de embriaguez	0	0	0.
92	viernes	h06	choque frontal excentrico	carigan	imprudencia del conductor	0	0	12.
84	lunes	h11	estrellamiento	san sebastian	cruce de animales en	0	0	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_causas_2021.pdf') as pdf:
    pdf.savefig(fig, bbox_inches='tight')
```

[illegible]