

Variable "dia"

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

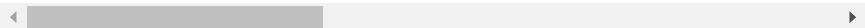
```
# -----
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.tree import export_graphviz
from sklearn.tree import export_text
from sklearn.model_selection import GridSearchCV
from sklearn.compose import 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.
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-pac

	fecha	dia	hora	latitud	longitud	tipologia	direccion	zona	parroqu
0	01-01	viernes	h11	-3.991993	-79.201155	estrellamiento	orillas del zamora y jose felix de valdivieso	urbana	
1	01-05	martes	h17	-4.020370	-79.217962	choque lateral perpendicular	benjamin carrion y gustavo serrano	urbana	
2	01-05	martes	h12	-3.987230	-79.202984	choque por alcance	nueva loja y guaranda	urbana	
3	01-08	viernes	h10	-3.989410	-79.236506	atropello	angel felicisimo rojas	urbana	
4	01-09	sabado	h06	-3.979784	-79.218689	choque lateral angular	isidro ayora y habana	urbana	



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()
```

```
/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
```

```
#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` argument
```

```
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` argument
```

```
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: DeprecationWarnin
`should_run_async` will not call `transform_cell` automatically in the future. Pleas
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fal
133	sabado	h19	choque por alcance	punzara	conducir en estado de embriaguez	0	
368	viernes	h20	choque lateral perpendicular	san sebastian	conducir en estado de embriaguez	1	
256	sabado	h01	choque lateral perpendicular	san sebastian	imprudencia del conductor	4	
131	jueves	h23	choque por alcance	el sagrario	conducir en estado de embriaguez	0	
244	jueves	h13	choque lateral perpendicular	sucre	imprudencia del conductor	0	

```
df.head()
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarnin
`should_run_async` will not call `transform_cell` automatically in the future. Pleas
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallec
0	viernes	h11	estrellamiento	el valle	conducir en estado de embriaguez	1	
1	martes	h17	choque lateral perpendicular	punzara	conducir en estado de embriaguez	1	
2	martes	h12	choque por	sucre	no mantener la distancia	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()
```

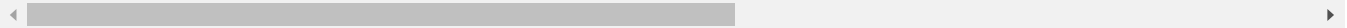
```
imprudencia del conductor      161
conducir en estado de embriaguez  88
conducir en exceso de velocidad  75
imprudencia del peaton         15
no respetar las senales de transito  15
fallas mecanicas no previsibles  6
no mantener la distancia reglamentaria  2
```

```
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
```

```
['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
```



```
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']
```

#Aplico seleccion de similitudes de tipologia para mejorar la precisión del modelo

```
df['tipologia']= df['tipologia'].apply(lambda x:
    1 if x == 'arrollamiento' else
    2 if x == 'atinico' 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
1 if x == 'h04' else
1 if x == 'h05' else
1 if x == 'h06' else
1 if x == 'h07' else
2 if x == 'h08' else
2 if x == 'h09' else
2 if x == 'h10' else
2 if x == 'h11' else
3 if x == 'h12' else
3 if x == 'h13' else
3 if x == 'h14' else
3 if x == 'h15' else
4 if x == 'h16' else
4 if x == 'h17' else
4 if x == 'h18' else
4 if x == 'h19' else
5 if x == 'h20' else
5 if x == 'h21' else
5 if x == 'h22' else
5)

```

```
df.head()
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos
0	2	2	8	3	1	1	0
1	1	4	5	5	1	1	0
2	1	3	5	4	10	0	0
3	2	2	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: DeprecationWarnin
```

```
`should_run_async` will not call `transform_cell` automatically in the future. Pleas
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos
312	2	0	5	2	1	0	0
26	1	5	5	1	4	1	0
89	1	5	8	3	1	0	0
102	2	0	8	4	1	0	0
41	1	2	9	3	12	1	0
226	3	2	5	4	2	1	0
339	2	0	8	3	2	0	0
354	3	4	6	3	2	0	0
61	1	3	3	1	2	1	0
224	2	2	5	1	7	2	0

Verificacion de correlacion de variables

```
#corelacion
```

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

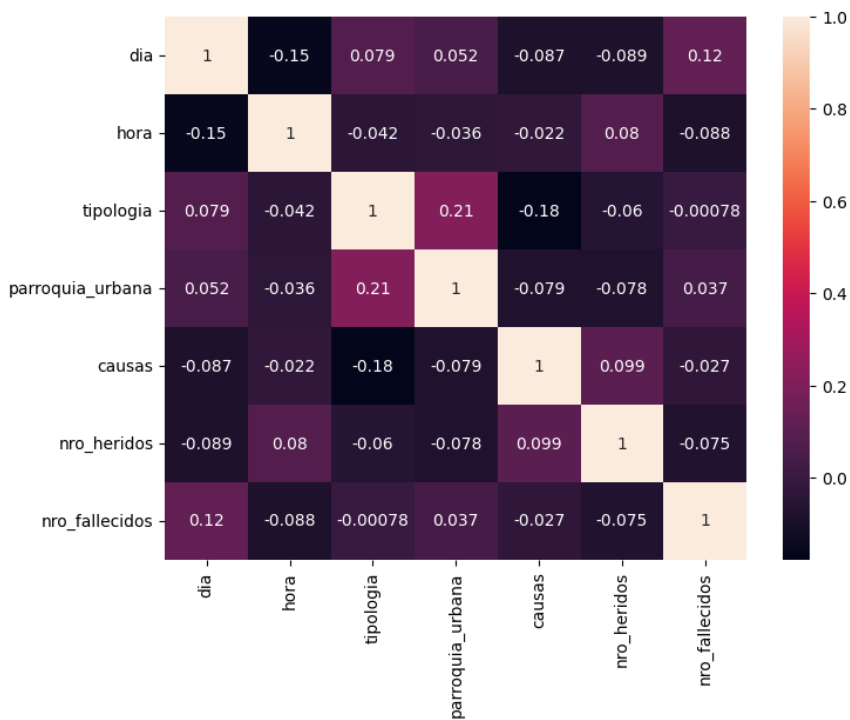
```
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(corr_df, annot=True)
```

```
plt.show()
```

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

```
`should_run_async` will not call `transform_cell` automatically in the future. Pleas
```



Seleccionamos la columna que vamos a predecir los datos

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

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

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

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

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

```
['hora', 'tipologia', '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()
```

	dia
0	2
1	1
2	1
3	2
4	3

Cargamos de modelo 2018-2020

```
from joblib import load
arbol_modelo = load('modelo_dia.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)
```

```
[3 3 1 1 3 1 3 3 3 1 3 1 3 1 3 3 3 1 3 3 3 1 3 3 3 3 3 3 1 3 3 3 3 3 3
 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3 1 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 1 3 1 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 3
 3 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 1 3 3 3 1 3 3 3
 1 3 3 1 3 3 3 1 3 1 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 1 3 1 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 3 3 3 3 3 1 3 3 3 3 3 3 1 3 3 3
 3 1 3 1 3 3 3 3 3 1 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 1 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3]
```

```
/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_dia_2021.xlsx")
```

	precision	recall	f1-score	support
1	0.413793	0.183206	0.253968	131.000000
2	0.000000	0.000000	0.000000	119.000000
3	0.339744	0.883333	0.490741	120.000000
accuracy	0.351351	0.351351	0.351351	0.351351
macro avg	0.251179	0.355513	0.248236	370.000000
weighted avg	0.256692	0.351351	0.249078	370.000000

```
/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()
```

Predicciones de los datos obtenidas

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

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarnin
`should_run_async` will not call `transform_cell` automatically in the future. Pleas
```

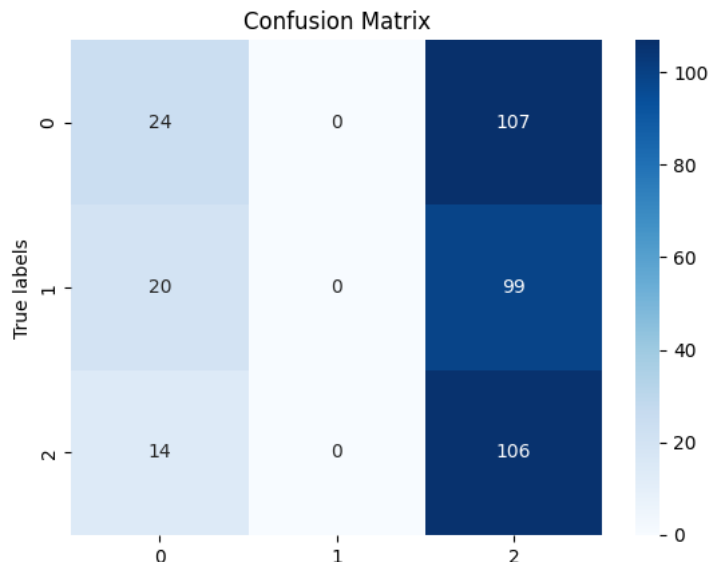
	0	1	2
0	3		
1	3		
2	1		
3	1		
4	3		

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



Concatenamos los datos de las horas originales con los datos predichos

```
#Para concatenar variables
import pandas as pd
```

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

	dia_prediccion	dia		
0	3	2		
1	3	1		
2	1	1		
3	1	2		
4	3	3		
5	1	3		
6	3	3		
7	3	3		
8	3	1		
9	1	1		

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: 35.13513513513514
Exhaustividad árboles de decisión: 35.13513513513514
Precisión árboles de decisión: 35.13513513513514
/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` argument

Transformamos las variables

```
#transformar datos para visualización de variables predictoras y originales
df_combined['dia']= df_combined['dia'].apply(lambda x:
                                             'lunes-miercoles' if x == 1 else
                                             'jueves-viernes' if x == 2 else
                                             'sabado-domingo')

df_combined['dia_prediccion']= df_combined['dia_prediccion'].apply(lambda x:
                                                                    'lunes-miercoles' if x == 1 else
                                                                    'jueves-viernes' if x == 2 else
                                                                    'sabado-domingo')

/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)
#ax[1].plot(df['dia'],df['hora'],color = 'tab:purple')
sns.countplot(x='dia',hue = 'dia', data = df_combined, ax = ax[1]) #Muestre el conteo de observaciones en cada contenedor categórico usa
ax[1].set_title('dia')

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_dia_2021.png")
```

Graficamos las estadísticas predecidas de los datos

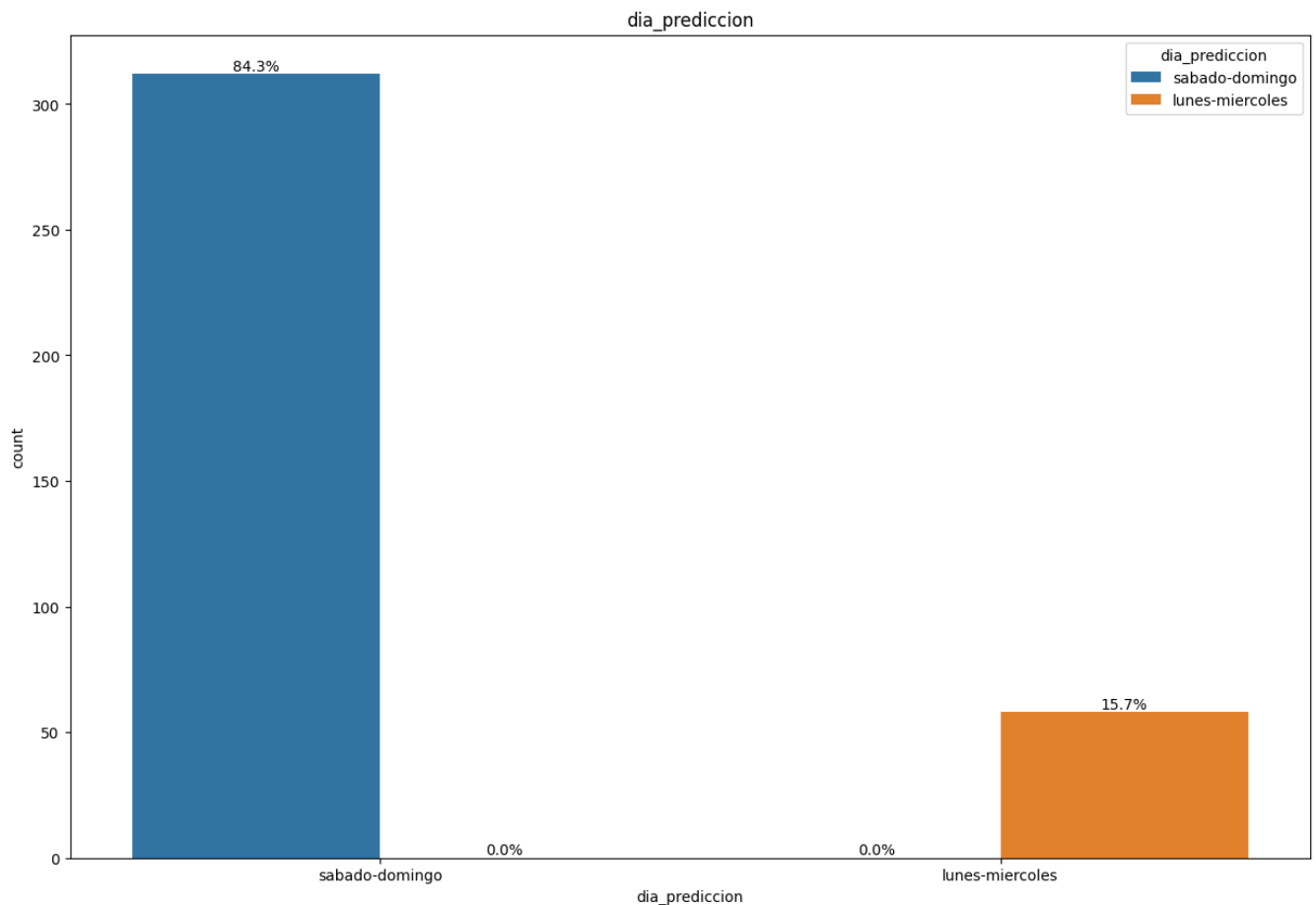
```
ax = plt.subplots(figsize = (15,10))
ncount=len(df_combined)
sns.countplot(x='dia_prediccion',hue = 'dia_prediccion', data = df_combined, ax = ax[1]) #Muestre el conteo de observaciones en cada con
ax[1].set_title('dia_prediccion')

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_dia_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 to `transformed_cell` argumer



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

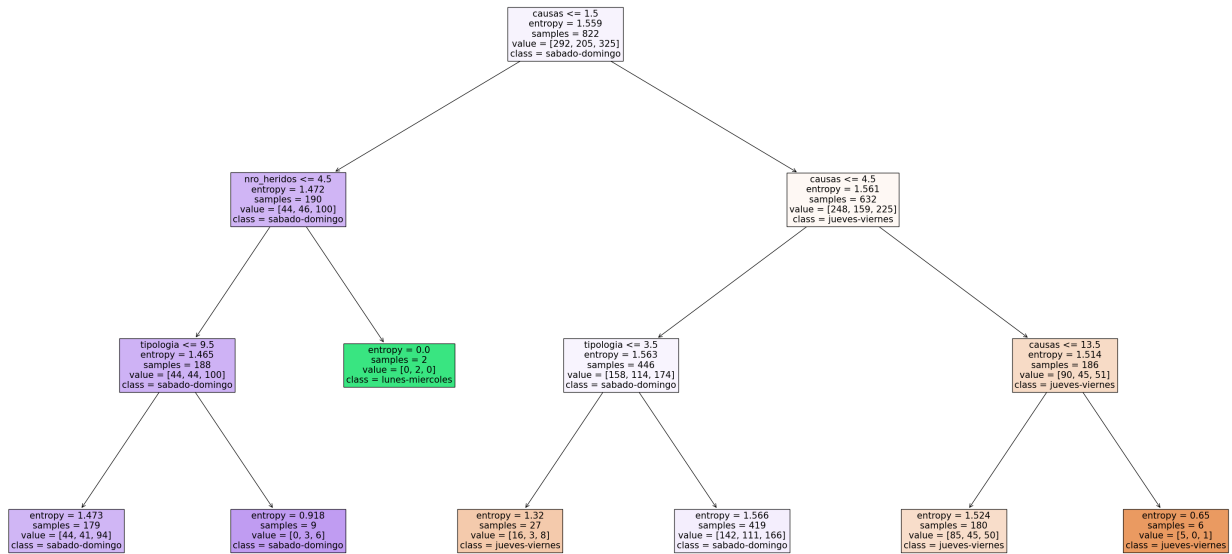
```
#target_names_str = [str(name) for name in target_names]
variable = df['dia'].apply(lambda x: 'lunes-miercoles' if x == 1 else
                          'jueves-viernes' if x == 2 else
                          'sabado-domingo')
claseVar = variable.unique().tolist()
target_names_str = [str(name) 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=(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_dia_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(type(y_proba.shape))
print(type(y_pred))
y_pred_valid = np.clip(y_pred, 0, y_proba.shape[1] - 1)

probabilidad_acierto = np.round(y_proba[np.arange(len(y_proba)), y_pred_valid] * 100, 2)

#probabilidad_acierto = np.round(y_proba[0, valid_indices[0][y_pred[valid_indices]]] * 100, 2)
#probabilidad_acierto = np.round(y_proba[np.arange(len(y_proba)), y_pred] * 100, 2)
#probabilidad_acierto = np.round(y_proba[0, y_pred] * 100, 2) #axis 1
#probabilidad_acierto = np.round(y_proba[0][y_pred] * 100, 2) #axis 0
print("Probabilidad de Acierto: " + str(probabilidad_acierto) + "%")

<class 'tuple'>
<class 'numpy.ndarray'>
Probabilidad de Acierto: [52.51 52.51 25. 25. 52.51 25. 39.62 39.62 39.62 25. 39.62 25.
52.51 25. 39.62 39.62 39.62 0. 39.62 52.51 39.62 0. 52.51 39.62
39.62 39.62 39.62 39.62 52.51 25. 39.62 52.51 39.62 39.62 52.51 52.51
39.62 39.62 39.62 39.62 39.62 25. 52.51 39.62 52.51 39.62 52.51 25.
39.62 66.67 52.51 39.62 52.51 39.62 25. 39.62 39.62 39.62 25. 39.62
39.62 11.11 52.51 39.62 52.51 39.62 39.62 39.62 25. 39.62 52.51 39.62
52.51 39.62 52.51 39.62 52.51 39.62 39.62 39.62 39.62 39.62 39.62
25. 39.62 11.11 39.62 39.62 52.51 39.62 52.51 39.62 52.51 0. 39.62
39.62 39.62 39.62 39.62 25. 39.62 52.51 52.51 39.62 11.11 39.62 39.62
39.62 39.62 39.62 39.62 39.62 52.51 39.62 52.51 39.62 25. 39.62 52.51
39.62 25. 39.62 39.62 39.62 39.62 39.62 39.62 25. 39.62 52.51
39.62 52.51 39.62 39.62 52.51 25. 39.62 39.62 39.62 25. 52.51 52.51
11.11 39.62 39.62 39.62 25. 39.62 39.62 11.11 39.62 39.62 39.62 11.11
39.62 25. 39.62 52.51 52.51 39.62 52.51 39.62 39.62 52.51 39.62 25.
39.62 39.62 52.51 39.62 52.51 25. 39.62 0. 39.62 39.62 39.62 39.62
39.62 39.62 39.62 52.51 39.62 52.51 39.62 52.51 52.51 52.51 39.62 39.62
39.62 39.62 39.62 39.62 39.62 39.62 39.62 52.51 39.62 25. 11.11
39.62 39.62 39.62 39.62 39.62 39.62 25. 39.62 52.51 52.51 52.51 52.51
52.51 52.51 25. 39.62 39.62 52.51 39.62 39.62 25. 52.51 39.62 39.62
39.62 39.62 39.62 39.62 39.62 39.62 52.51 25. 39.62 39.62 66.67 11.11 39.62
25. 25. 39.62 52.51 39.62 39.62 25. 39.62 39.62 25. 39.62 11.11
25. 39.62 39.62 52.51 39.62 39.62 39.62 52.51 11.11 39.62 11.11 39.62
39.62 39.62 39.62 25. 39.62 25. 39.62 52.51 39.62 11.11 39.62 39.62
39.62 39.62 39.62 39.62 52.51 39.62 39.62 25. 39.62 39.62 52.51 52.51
```

```

39.62 39.62 52.51 52.51 52.51 39.62 52.51 39.62 39.62 52.51 39.62 39.62
52.51 52.51 39.62 52.51 52.51 39.62 52.51 39.62 52.51 39.62 39.62 39.62
52.51 39.62 52.51 25. 39.62 52.51 52.51 39.62 39.62 25. 39.62 52.51
39.62 39.62 39.62 39.62 39.62 39.62 39.62 39.62 39.62 52.51 25. 39.62
52.51 52.51 39.62 39.62 39.62 39.62 39.62 39.62 39.62 39.62 39.62 11.11
39.62 39.62 39.62 39.62 25. 39.62 39.62 39.62 39.62 52.51 39.62 11.11
39.62 52.51 52.51 39.62 39.62 39.62 39.62 39.62 52.51 52.51]%
```

```

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

```

<class 'numpy.ndarray'>
  probabilidad
0      52.51
1      52.51
2      25.00
3      25.00
4      52.51
..      ...
365     39.62
366     39.62
367     39.62
368     52.51
369     52.51
```

```
[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 to `transformed_cell` argumer
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos	probabilidad	predicciones
234	3	2	5	4	7	1	0	25.00	1
84	2	5	5	2	7	0	0	25.00	1
272	2	3	5	4	2	0	0	39.62	3
130	2	1	8	4	2	0	0	39.62	3
266	1	4	9	4	4	0	0	39.62	3
306	1	4	8	5	1	0	0	52.51	3
248	1	5	10	4	2	1	0	39.62	3
37	2	5	8	3	4	0	0	39.62	3
192	1	4	8	1	2	0	0	39.62	3
344	1	0	5	2	2	0	0	39.62	3

```

probabilidades_2021_df['predicciones'] = probabilidades_2021_df['predicciones'].apply(lambda x:
    'lunes-miercoles' if x == 1 else
    'jueves-viernes' if x == 2 else
    'sabado-domingo')
```

```

probabilidades_2021_df['dia'] = probabilidades_2021_df['dia'].apply(lambda x:
    'lunes-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-h03' if x == 0 else
    'h04-h07' if x == 1 else
    'h08-h11' if x == 2 else
    'h12-h15' if x == 3 else
```

```
'h16-h19' if x == 4 else
'h20-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 to `transformed_cell` argumer
```

	dia	hora	tipologia	parroquia_urbana	causas	nro_heridos	nro_fallecidos	probabilidad	predicciones
5	sabado-domingo	h08-h11	choque	sucre	no mantener la distancia reglamentaria	0	0	25.00	lunes-miercoles
171	sabado-domingo	h20-h23	choque	san sebastian	imprudencia del conductor	0	0	39.62	sabado-domingo
360	lunes-miercoles	h16-h19	roce	el sagrario	imprudencia del conductor	0	0	39.62	sabado-domingo
50	sabado-domingo	h04-h07	estrellamiento	sucre	conducir en estado de embriaguez	0	0	52.51	sabado-domingo
263	jueves-viernes	h08-h11	choque	sucre	imprudencia del conductor	0	0	39.62	sabado-domingo
192	lunes-miercoles	h16-h19	estrellamiento	el sagrario	imprudencia del conductor	0	0	39.62	sabado-domingo
366	lunes-miercoles	h12-h15	choque	carigan	imprudencia del conductor	0	0	39.62	sabado-domingo
265	lunes-miercoles	h08-h11	roce	sucre	conducir en exceso de velocidad	0	0	39.62	sabado-domingo
195	sabado-domingo	h12-h15	choque	punzara	imprudencia del conductor	0	0	39.62	sabado-domingo
89	lunes-miercoles	h20-h23	estrellamiento	el valle	conducir en estado de embriaguez	0	0	52.51	sabado-domingo

```
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_dia_2021.pdf') as pdf:
    pdf.savefig(fig, bbox_inches='tight')
```

[illegible]