CALCULO DE TIEMPO ESTIMADO DE LA RUTA

El presente modelo se realiza para cálcular el tiempo estimado basado en los valores de carga de tráfico predicha por el modelo de tráfico.

0.Inicialización

```
import numpy as np
import pandas as pd
import datetime as dt
```

1.Predicción sobre la ruta

Carga del fichero de carga en puntos de medida predichos y carga del equivalente de ruta según los puntos de medida

```
carga_predicha=pd.read_csv('/content/drive/MyDrive/Proyecto PSIA/Notebooks/prediccion_c
carga_predicha=carga_predicha.drop(['Unnamed: 0', 'carga', 'latitud', 'longitud', 'LLUV
       'hora_ok_09:00', 'hora_ok_09:15', 'hora_ok_09:30', 'DIA_OK_jueves',
       'DIA_OK_lunes', 'DIA_OK_martes', 'DIA_OK_miercoles', 'DIA_OK_viernes',
       'Semana_mes_1', 'Semana_mes_2', 'Semana_mes_3', 'Semana_mes_4',
       'Semana_mes_5', 'tipo_via_carga_alta', 'tipo_via_carga_baja',
       'tipo_via_carga_media', 'tipo_via_carga_muy_alta'], axis=1)
carga_predicha.head()

```

	id	cluster	carga_final
0	5257	35.0	18.0
1	7029	283.0	30.0
2	6644	68.0	94.0
3	6288	282.0	38.0
4	7024	63.0	87.0

```
ruta_equivalente=pd.read_csv('/content/drive/MyDrive/Proyecto PSIA/Notebooks/Ptos_segur

ruta_equivalente[['id','longitud','latitud']]=ruta_equivalente[['id_x','longitud_x','la
ruta_equivalente=ruta_equivalente.drop(['id_x','longitud_x','latitud_x','Unnamed: 0','c
ruta_equivalente.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 50 entries, 0 to 49
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   repeticiones  50 non-null     int64  
 1   id            50 non-null     int64  
 2   longitud      50 non-null     float64
 3   latitud       50 non-null     float64
dtypes: float64(2), int64(2)
memory usage: 1.7 KB
```

Se añade la carga predicha a los puntos de medida de la ruta

```
carga_ruta=ruta_equivalente.merge(carga_predicha,on='id',how='left')
carga_ruta.head()
```

	repeticiones	id	longitud	latitud	cluster	carga_final
0	5	3511	-3.710894	40.439773	164.0	81.0
1	4	3619	-3.713161	40.435190	48.0	57.0
2	5	3621	-3.714337	40.429745	48.0	44.0
3	4	4284	-3.711724	40.424183	9.0	25.0
4	7	4285	-3.711466	40.423156	9.0	45.0

2.Calculo de la estimación

Comprobamos los valores nulos y los eliminamos

```
carga_ruta_null=carga_ruta[carga_ruta.carga_final.isnull()]
carga_ruta=carga_ruta.dropna()
```

Calculamos la carga media. Multiplicamos el número de repeticiones por la carga final, sumamos todos los valores y dividimos por el número total de repeticiones de la ruta

```
carga_ruta['carga_total']=carga_ruta['carga_final']*carga_ruta['repeticiones']
carga_ruta['objetivo']='sumarizar'
carga_ruta.head()
```

	repeticiones	id	longitud	latitud	cluster	carga_final	carga_total	objet

```
carga_ruta_total=carga_ruta.groupby(by='objetivo').sum()
carga_ruta_total
```

objetivo	repeticiones	id	longitud	latitud	cluster	carga_final	carga
sumarizar	180	241836	-170.231149	1860.381302	5349.0	2190.0	

```
carga_ruta_total['carga_media']=carga_ruta_total['carga_total']/carga_ruta_total['repet
carga_ruta_total
```

objetivo	repeticiones	id	longitud	latitud	cluster	carga_final	carga
sumarizar	180	241836	-170.231149	1860.381302	5349.0	2190.0	

Corregimos el tiempo estimado por Google maps en función de la carga de tráfico predicha:

- Muy baja 0-20 --> -10%
- Baja 20-40 --> +0%
- Media 40-60 --> +10%
- Alta 60-80 --> +15%
- Muy alta 80-100 --> +20%

```
def carga_cat(carga_ruta_total) :

    if carga_ruta_total["carga_media"] <= 20 :
        return -0.1
    elif (carga_ruta_total["carga_media"] > 20) & (carga_ruta_total["carga_media"] <= 4
        return 0
    elif (carga_ruta_total["carga_media"] > 40) & (carga_ruta_total["carga_media"] <= 6
        return 0.1
    elif (carga_ruta_total["carga_media"] > 60) & (carga_ruta_total["carga_media"] <= 8
        return 0.15
    elif carga_ruta_total["carga_media"] > 80 :
        return 0.2
carga_ruta_total["Factor_corr"] = carga_ruta_total.apply(lambda carga_ruta_total:carga_
                                  axis = 1)
carga_ruta_total.loc[:,'Factor_corr']
carga_ruta_total.head()
```

objetivo	repeticiones	id	longitud	latitud	cluster	carga_final	carga
sumarizar	180	241836	-170.231149	1860.381302	5349.0	2190.0	

3.Resultado tiempo estimado Ruta

Aplicamos el factor de corrección al tiempo estimado por google maps: 21 minutos

```
carga_ruta_total['tiempo_google']=21
carga_ruta_total['tiempo_corregido']=carga_ruta_total['tiempo_google']*(1+carga_ruta_tc
carga_ruta_total
tiempo_google=carga_ruta_total['tiempo_google'].to_numpy()
tiempo_estimado=carga_ruta_total['tiempo_corregido'].to_numpy()
datetime=dt
tiempo=str(datetime.timedelta(minutes=tiempo_estimado[0]))

print('Según Google Maps el tiempo estimado de llegada entre Plaza Castilla y Plaza Esp
print('Teniendo en cuenta la previsión del tráfico actual según las estimaciones del Ay
print("\033[1;35m", tiempo.split(':')[1], end=' ')
print("\033[0;30m" + 'minutos', end='')
print("\033[1;35m", tiempo.split(':')[2], end=' ')
print("\033[0;30m" + 'segundos')

```

```
Según Google Maps el tiempo estimado de llegada entre Plaza Castilla y Plaza España
Teniendo en cuenta la previsión del tráfico actual según las estimaciones del Ayunta
```