Parte1_proyecto_adp

August 15, 2025

[ ]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```python
df=pd.read_csv('Scooter_Trips_2020.csv')
df
```

[ ]:
```
              Date  Hour  Trip Distance  Trip Duration Vendor  \
0       08/12/2020     5              5             21   spin
1       08/12/2020     7             13            101   spin
2       08/12/2020     7              7             50   bird
3       08/12/2020     7           3815            840   spin
4       08/12/2020     8           1444            445   spin
…              …    …             …            …     …
157289  12/12/2020    21            335            186   lime
157290  12/12/2020    21           2704           1254   lime
157291  12/12/2020    21           9257           2214   spin
157292  12/12/2020    21            878            325   lime
157293  12/12/2020    21            490            212   lime

        Start Community Area Number  End Community Area Number  \
0                              31.0                       31.0
1                               7.0                        7.0
2                              77.0                       77.0
3                               6.0                        3.0
4                               3.0                        6.0
…                               …                        …
157289                         23.0                       23.0
157290                         37.0                       61.0
157291                          6.0                        6.0
157292                         28.0                       24.0
157293                          8.0                        8.0

       Start Community Area Name End Community Area Name  \
0                 LOWER WEST SIDE         LOWER WEST SIDE
1                    LINCOLN PARK            LINCOLN PARK
```

```
2                    EDGEWATER            EDGEWATER
3                    LAKE VIEW               UPTOWN
4                       UPTOWN            LAKE VIEW
…                            …                    …
157289           HUMBOLDT PARK        HUMBOLDT PARK
157290             FULLER PARK             NEW CITY
157291               LAKE VIEW            LAKE VIEW
157292          NEAR WEST SIDE            WEST TOWN
157293         NEAR NORTH SIDE      NEAR NORTH SIDE

        Start Centroid Latitude  Start Centroid Longitude  \
0                     41.848335                -87.675179
1                     41.921880                -87.645647
2                     41.987114                -87.664343
3                     41.943514                -87.657498
4                     41.965435                -87.655145
…                             …                         …
157289                41.900813                -87.723955
157290                41.813368                -87.632599
157291                41.943514                -87.657498
157292                41.874254                -87.664619
157293                41.899528                -87.633571

        End Centroid Latitude  End Centroid Longitude
0                   41.848335              -87.675179
1                   41.921880              -87.645647
2                   41.987114              -87.664343
3                   41.965435              -87.655145
4                   41.943514              -87.657498
…                           …                       …
157289              41.900813              -87.723955
157290              41.808705              -87.657612
157291              41.943514              -87.657498
157292              41.901459              -87.675568
157293              41.899528              -87.633571

[157294 rows x 13 columns]
```

Viajes de agosto a diciembre 2020; distancias de 5-9257m, duraciones de 21-2214s; proveedores como 'spin', 'bird', 'lime'. Uso urbano variado.

[ ]:
```
df.isna().sum()
```

[ ]:
```
Date                    0
Hour                    0
Trip Distance           0
Trip Duration           0
```

```
Vendor                        0
Start Community Area Number   0
End Community Area Number     0
Start Community Area Name     0
End Community Area Name       0
Start Centroid Latitude       0
Start Centroid Longitude      0
End Centroid Latitude         0
End Centroid Longitude        0
dtype: int64
```

[ ]:
```
df
```

[ ]:
```
              Date  Hour  Trip Distance  Trip Duration Vendor  \
0       08/12/2020     5              5             21   spin
1       08/12/2020     7             13            101   spin
2       08/12/2020     7              7             50   bird
3       08/12/2020     7           3815            840   spin
4       08/12/2020     8           1444            445   spin
…              …    …             …              …     …
157289  12/12/2020    21            335            186   lime
157290  12/12/2020    21           2704           1254   lime
157291  12/12/2020    21           9257           2214   spin
157292  12/12/2020    21            878            325   lime
157293  12/12/2020    21            490            212   lime

        Start Community Area Number  End Community Area Number  \
0                              31.0                       31.0
1                               7.0                        7.0
2                              77.0                       77.0
3                               6.0                        3.0
4                               3.0                        6.0
…                                 …                         …
157289                         23.0                       23.0
157290                         37.0                       61.0
157291                          6.0                        6.0
157292                         28.0                       24.0
157293                          8.0                        8.0

       Start Community Area Name End Community Area Name  \
0                 LOWER WEST SIDE         LOWER WEST SIDE
1                    LINCOLN PARK            LINCOLN PARK
2                       EDGEWATER               EDGEWATER
3                       LAKE VIEW                  UPTOWN
4                          UPTOWN               LAKE VIEW
…                             …                       …
157289             HUMBOLDT PARK           HUMBOLDT PARK
```

```
157290             FULLER PARK            NEW CITY
157291               LAKE VIEW           LAKE VIEW
157292         NEAR WEST SIDE            WEST TOWN
157293        NEAR NORTH SIDE      NEAR NORTH SIDE

        Start Centroid Latitude  Start Centroid Longitude  \
0                     41.848335                -87.675179
1                     41.921880                -87.645647
2                     41.987114                -87.664343
3                     41.943514                -87.657498
4                     41.965435                -87.655145
…                           …                        …
157289                41.900813                -87.723955
157290                41.813368                -87.632599
157291                41.943514                -87.657498
157292                41.874254                -87.664619
157293                41.899528                -87.633571

        End Centroid Latitude  End Centroid Longitude
0                   41.848335              -87.675179
1                   41.921880              -87.645647
2                   41.987114              -87.664343
3                   41.965435              -87.655145
4                   41.943514              -87.657498
…                         …                      …
157289              41.900813              -87.723955
157290              41.808705              -87.657612
157291              41.943514              -87.657498
157292              41.901459              -87.675568
157293              41.899528              -87.633571

[157294 rows x 13 columns]
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157294 entries, 0 to 157293
Data columns (total 13 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   Date                         157294 non-null  object
 1   Hour                         157294 non-null  int64
 2   Trip Distance                157294 non-null  int64
 3   Trip Duration                157294 non-null  int64
 4   Vendor                       157294 non-null  object
 5   Start Community Area Number  157294 non-null  float64
 6   End Community Area Number    157294 non-null  float64
```

```
 7   Start Community Area Name    157294 non-null  object
 8   End Community Area Name      157294 non-null  object
 9   Start Centroid Latitude      157294 non-null  float64
 10  Start Centroid Longitude     157294 non-null  float64
 11  End Centroid Latitude        157294 non-null  float64
 12  End Centroid Longitude       157294 non-null  float64
dtypes: float64(6), int64(3), object(4)
memory usage: 15.6+ MB
```

[ ]:
```
# Convertir Date y Hour a una columna datetime

df['Dia_hora']=pd.to_datetime(df['Date']+' '+df['Hour'].astype(str)+ ':00:00',␣
 ↪format='%m/%d/%Y %H:%M:%S')
```

Facilita análisis temporal.

[ ]:
```
df
```

[ ]:
```
              Date  Hour  Trip Distance  Trip Duration Vendor  \
0       08/12/2020     5              5             21   spin
1       08/12/2020     7             13            101   spin
2       08/12/2020     7              7             50   bird
3       08/12/2020     7           3815            840   spin
4       08/12/2020     8           1444            445   spin
…              …    …            …             …    …
157289  12/12/2020    21            335            186   lime
157290  12/12/2020    21           2704           1254   lime
157291  12/12/2020    21           9257           2214   spin
157292  12/12/2020    21            878            325   lime
157293  12/12/2020    21            490            212   lime

        Start Community Area Number  End Community Area Number  \
0                              31.0                       31.0
1                               7.0                        7.0
2                              77.0                       77.0
3                               6.0                        3.0
4                               3.0                        6.0
…                              …                          …
157289                         23.0                       23.0
157290                         37.0                       61.0
157291                          6.0                        6.0
157292                         28.0                       24.0
157293                          8.0                        8.0

       Start Community Area Name End Community Area Name  \
0                LOWER WEST SIDE         LOWER WEST SIDE
1                   LINCOLN PARK            LINCOLN PARK
2                      EDGEWATER               EDGEWATER
```

```
3                      LAKE VIEW                UPTOWN
4                         UPTOWN             LAKE VIEW
…                              …                     …
157289             HUMBOLDT PARK         HUMBOLDT PARK
157290                FULLER PARK              NEW CITY
157291                  LAKE VIEW             LAKE VIEW
157292            NEAR WEST SIDE             WEST TOWN
157293           NEAR NORTH SIDE       NEAR NORTH SIDE

        Start Centroid Latitude  Start Centroid Longitude  \
0                     41.848335                -87.675179
1                     41.921880                -87.645647
2                     41.987114                -87.664343
3                     41.943514                -87.657498
4                     41.965435                -87.655145
…                             …                         …
157289                41.900813                -87.723955
157290                41.813368                -87.632599
157291                41.943514                -87.657498
157292                41.874254                -87.664619
157293                41.899528                -87.633571

        End Centroid Latitude  End Centroid Longitude            Dia_hora
0                   41.848335              -87.675179 2020-08-12 05:00:00
1                   41.921880              -87.645647 2020-08-12 07:00:00
2                   41.987114              -87.664343 2020-08-12 07:00:00
3                   41.965435              -87.655145 2020-08-12 07:00:00
4                   41.943514              -87.657498 2020-08-12 08:00:00
…                           …                      …                   …
157289              41.900813              -87.723955 2020-12-12 21:00:00
157290              41.808705              -87.657612 2020-12-12 21:00:00
157291              41.943514              -87.657498 2020-12-12 21:00:00
157292              41.901459              -87.675568 2020-12-12 21:00:00
157293              41.899528              -87.633571 2020-12-12 21:00:00

[157294 rows x 14 columns]
```

[ ]:
```python
# Crear variables: Dia de la semana, mes y hora del dia

df['Dia_semana']=df['Dia_hora'].dt.day_name()
df['Mes']=df['Dia_hora'].dt.month
df['Hora_dia']=df['Dia_hora'].dt.hour
```

Para agrupaciones (e.g., fines de semana).

[ ]:
```python
df
```

[ ]:
```
             Date  Hour  Trip Distance  Trip Duration Vendor  \
0       08/12/2020     5              5             21   spin
1       08/12/2020     7             13            101   spin
2       08/12/2020     7              7             50   bird
3       08/12/2020     7           3815            840   spin
4       08/12/2020     8           1444            445   spin
…              …   …             …             …    …
157289  12/12/2020    21            335            186   lime
157290  12/12/2020    21           2704           1254   lime
157291  12/12/2020    21           9257           2214   spin
157292  12/12/2020    21            878            325   lime
157293  12/12/2020    21            490            212   lime

        Start Community Area Number  End Community Area Number  \
0                              31.0                       31.0
1                               7.0                        7.0
2                              77.0                       77.0
3                               6.0                        3.0
4                               3.0                        6.0
…                               …                       …
157289                         23.0                       23.0
157290                         37.0                       61.0
157291                          6.0                        6.0
157292                         28.0                       24.0
157293                          8.0                        8.0

       Start Community Area Name End Community Area Name  \
0                 LOWER WEST SIDE         LOWER WEST SIDE
1                    LINCOLN PARK            LINCOLN PARK
2                      EDGEWATER               EDGEWATER
3                      LAKE VIEW                  UPTOWN
4                         UPTOWN               LAKE VIEW
…                            …                     …
157289             HUMBOLDT PARK           HUMBOLDT PARK
157290               FULLER PARK                NEW CITY
157291                 LAKE VIEW               LAKE VIEW
157292            NEAR WEST SIDE               WEST TOWN
157293           NEAR NORTH SIDE         NEAR NORTH SIDE

        Start Centroid Latitude  Start Centroid Longitude  \
0                     41.848335                -87.675179
1                     41.921880                -87.645647
2                     41.987114                -87.664343
3                     41.943514                -87.657498
4                     41.965435                -87.655145
…                          …                       …
157289                41.900813                -87.723955
```

```
157290                41.813368                 -87.632599
157291                41.943514                 -87.657498
157292                41.874254                 -87.664619
157293                41.899528                 -87.633571

        End Centroid Latitude  End Centroid Longitude            Dia_hora  \
0                   41.848335              -87.675179 2020-08-12 05:00:00
1                   41.921880              -87.645647 2020-08-12 07:00:00
2                   41.987114              -87.664343 2020-08-12 07:00:00
3                   41.965435              -87.655145 2020-08-12 07:00:00
4                   41.943514              -87.657498 2020-08-12 08:00:00
…                         …                       …                   …
157289              41.900813              -87.723955 2020-12-12 21:00:00
157290              41.808705              -87.657612 2020-12-12 21:00:00
157291              41.943514              -87.657498 2020-12-12 21:00:00
157292              41.901459              -87.675568 2020-12-12 21:00:00
157293              41.899528              -87.633571 2020-12-12 21:00:00

       Dia_semana  Mes  Hora_dia
0       Wednesday    8         5
1       Wednesday    8         7
2       Wednesday    8         7
3       Wednesday    8         7
4       Wednesday    8         8
…             …  …         …
157289   Saturday   12        21
157290   Saturday   12        21
157291   Saturday   12        21
157292   Saturday   12        21
157293   Saturday   12        21

[157294 rows x 17 columns]
```

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157294 entries, 0 to 157293
Data columns (total 17 columns):
 #   Column                     Non-Null Count   Dtype
---  ------                     --------------   -----
 0   Date                       157294 non-null  object
 1   Hour                       157294 non-null  int64
 2   Trip Distance              157294 non-null  int64
 3   Trip Duration              157294 non-null  int64
 4   Vendor                     157294 non-null  object
 5   Start Community Area Number 157294 non-null  float64
 6   End Community Area Number  157294 non-null  float64
```

```
 7   Start Community Area Name    157294 non-null  object
 8   End Community Area Name      157294 non-null  object
 9   Start Centroid Latitude      157294 non-null  float64
 10  Start Centroid Longitude     157294 non-null  float64
 11  End Centroid Latitude        157294 non-null  float64
 12  End Centroid Longitude       157294 non-null  float64
 13  Dia_hora                     157294 non-null  datetime64[ns]
 14  Dia_semana                   157294 non-null  object
 15  Mes                          157294 non-null  int32
 16  Hora_dia                     157294 non-null  int32
dtypes: datetime64[ns](1), float64(6), int32(2), int64(3), object(5)
memory usage: 19.2+ MB
```

[ ]:
```
df['Trip Distance'] = pd.to_numeric(df['Trip Distance'], errors='coerce')
df['Trip Duration'] = pd.to_numeric(df['Trip Duration'], errors='coerce')

df['Trip Distance (km)'] = df['Trip Distance'] / 1000 # convertimos a kilometros

# umbrales del percentil 99
dist_limit = df['Trip Distance (km)'].quantile(0.99)
dur_limit = df['Trip Duration'].quantile(0.99)

# Filtrmos outliers
df_filtered = df[
    (df['Trip Distance (km)'] <= dist_limit) &
    (df['Trip Duration'] <= dur_limit)
]

print(f"Total original: {df.shape[0]}")
print(f"Total filtrado: {df_filtered.shape[0]}")
```

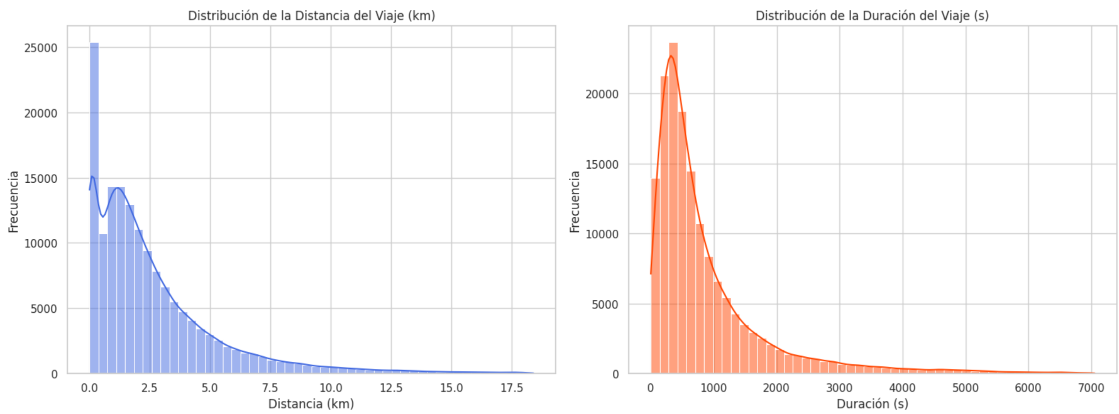
```
Total original: 157294
Total filtrado: 154585
```

Elimina extremos para robustez.

[ ]:
```
sns.set(style="whitegrid")
plt.figure(figsize=(16, 6))
fig, axes = plt.subplots(1, 2, figsize=(16, 6))


sns.histplot(df_filtered['Trip Distance (km)'], bins=50, kde=True, ax=axes[0],
 ↪color='royalblue')
axes[0].set_title('Distribución de la Distancia del Viaje (km)')
axes[0].set_xlabel('Distancia (km)')
axes[0].set_ylabel('Frecuencia')
```

```
sns.histplot(df_filtered['Trip Duration'], bins=50, kde=True, ax=axes[1],␣
 ↪color='orangered')
axes[1].set_title('Distribución de la Duración del Viaje (s)')
axes[1].set_xlabel('Duración (s)')
axes[1].set_ylabel('Frecuencia')

plt.tight_layout()
plt.show()
```

```
<Figure size 1600x600 with 0 Axes>
```



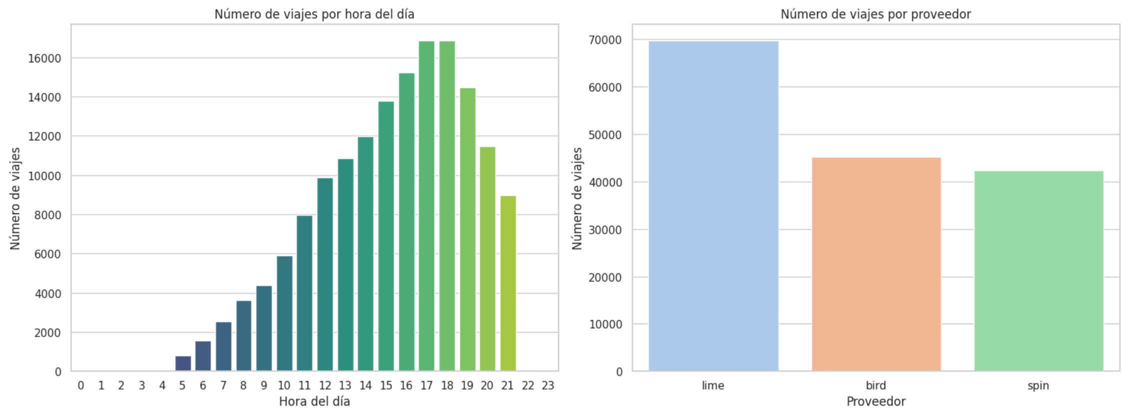
Distancia: Pico <2km, cola larga. Duración: <1000s mayoritario. Viajes cortos/rápidos.

[ ]:
```
# viajes por hora y por provedor

trips_by_hour = df.groupby('Hour').size()

trips_by_vendor = df['Vendor'].value_counts()

sns.set(style="whitegrid")
plt.figure(figsize=(16, 6))

fig, axes = plt.subplots(1, 2, figsize=(16, 6))

sns.barplot(x=trips_by_hour.index, y=trips_by_hour.values, ax=axes[0],␣
 ↪palette='viridis')
axes[0].set_title('Número de viajes por hora del día')
axes[0].set_xlabel('Hora del día')
axes[0].set_ylabel('Número de viajes')

sns.barplot(x=trips_by_vendor.index, y=trips_by_vendor.values, ax=axes[1],␣
 ↪palette='pastel')
axes[1].set_title('Número de viajes por proveedor')
```

```
axes[1].set_xlabel('Proveedor')
axes[1].set_ylabel('Número de viajes')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2819315463.py:12: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=trips_by_hour.index, y=trips_by_hour.values, ax=axes[0],
palette='viridis')
/tmp/ipython-input-2819315463.py:17: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=trips_by_vendor.index, y=trips_by_vendor.values, ax=axes[1],
palette='pastel')
```

```
<Figure size 1600x600 with 0 Axes>
```



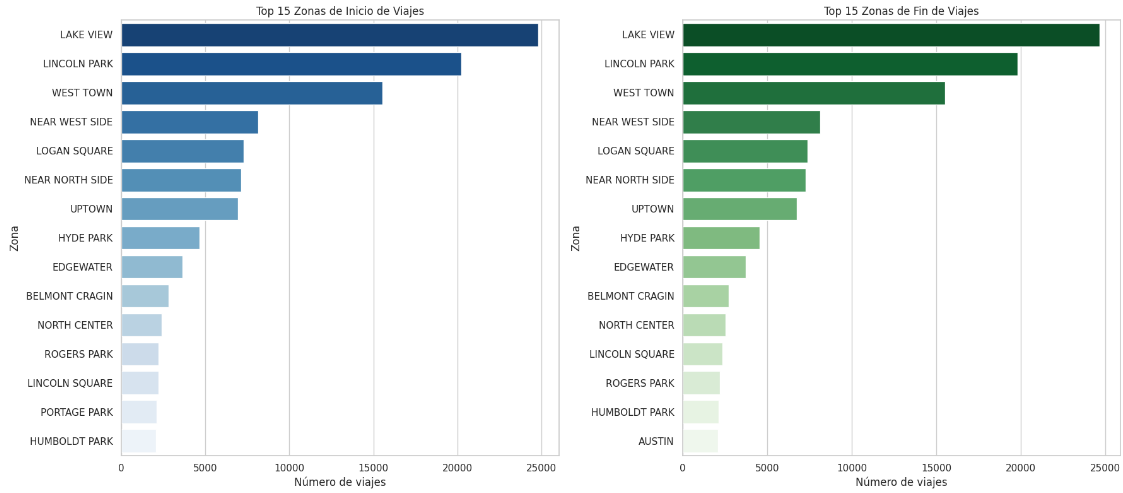
Hora: Pico 12-18h. Proveedor: Uno domina (e.g., 'lime'). Uso diurno.

[ ]:
```
df_clean = df.copy()

start_areas = df_clean['Start Community Area Name'].value_counts().
 ↪sort_values(ascending=False)
end_areas = df_clean['End Community Area Name'].value_counts().
 ↪sort_values(ascending=False)
```

```python
plt.figure(figsize=(18, 8))
fig, axes = plt.subplots(1, 2, figsize=(18, 8))

sns.barplot(y=start_areas.index[:15], x=start_areas.values[:15], ax=axes[0],␣
 ↪palette='Blues_r')
axes[0].set_title('Top 15 Zonas de Inicio de Viajes')
axes[0].set_xlabel('Número de viajes')
axes[0].set_ylabel('Zona')

sns.barplot(y=end_areas.index[:15], x=end_areas.values[:15], ax=axes[1],␣
 ↪palette='Greens_r')
axes[1].set_title('Top 15 Zonas de Fin de Viajes')
axes[1].set_xlabel('Número de viajes')
axes[1].set_ylabel('Zona')

plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-2819165702.py:10: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=start_areas.index[:15], x=start_areas.values[:15], ax=axes[0],
palette='Blues_r')
/tmp/ipython-input-2819165702.py:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(y=end_areas.index[:15], x=end_areas.values[:15], ax=axes[1],
palette='Greens_r')
```

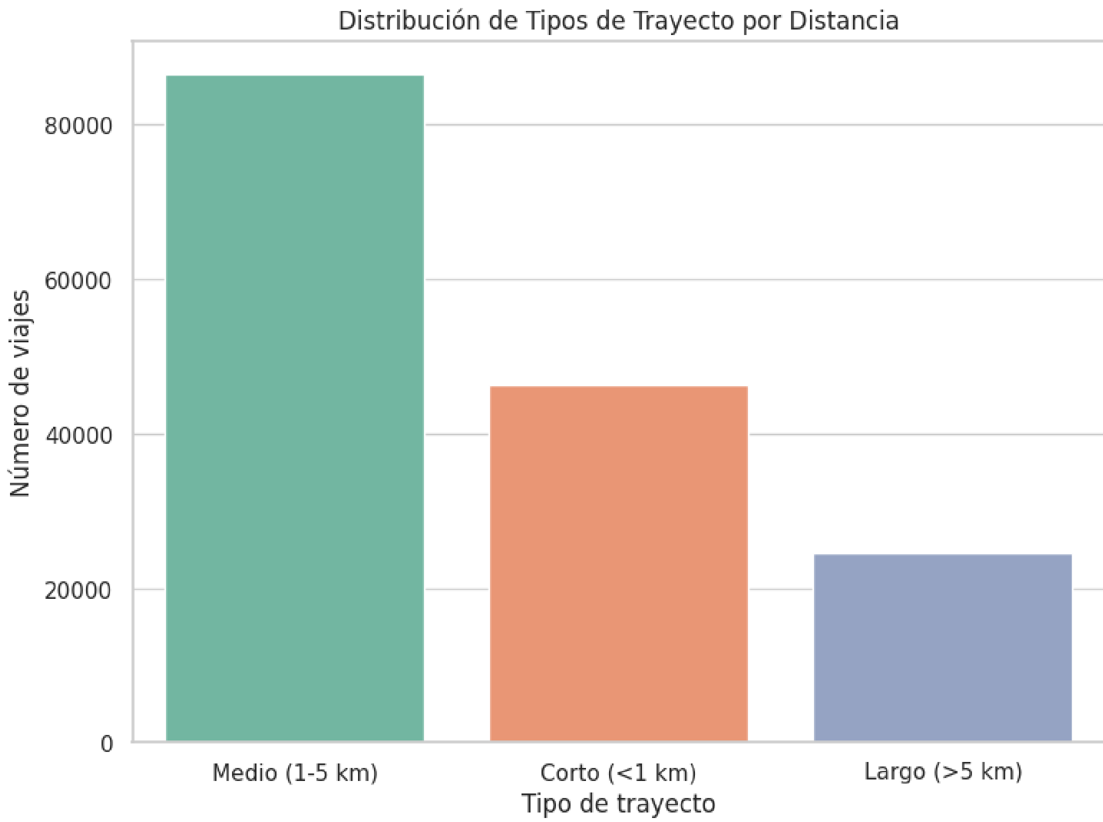
```
<Figure size 1800x800 with 0 Axes>
```



Lake View/Lincoln Park top (>20k). Áreas centrales populares; similar start/end (circulares?).

[ ]:
```
df_clean['Trip Distance (km)'] = df_clean['Trip Distance'] / 1000 # convertimos␣
↪a kilometros


def clasificar_trayecto(distancia):
    if distancia < 1:
        return 'Corto (<1 km)'
    elif distancia <= 5:
        return 'Medio (1-5 km)'
    else:
        return 'Largo (>5 km)'


df_clean['Tipo de Trayecto'] = df_clean['Trip Distance (km)'].
↪apply(clasificar_trayecto)


trayectos_tipo = df_clean['Tipo de Trayecto'].value_counts()


sns.set(style="whitegrid")
plt.figure(figsize=(8, 6))
sns.barplot(x=trayectos_tipo.index, y=trayectos_tipo.values, palette='Set2')
plt.title('Distribución de Tipos de Trayecto por Distancia')
plt.xlabel('Tipo de trayecto')
plt.ylabel('Número de viajes')
plt.tight_layout()
plt.show()
```

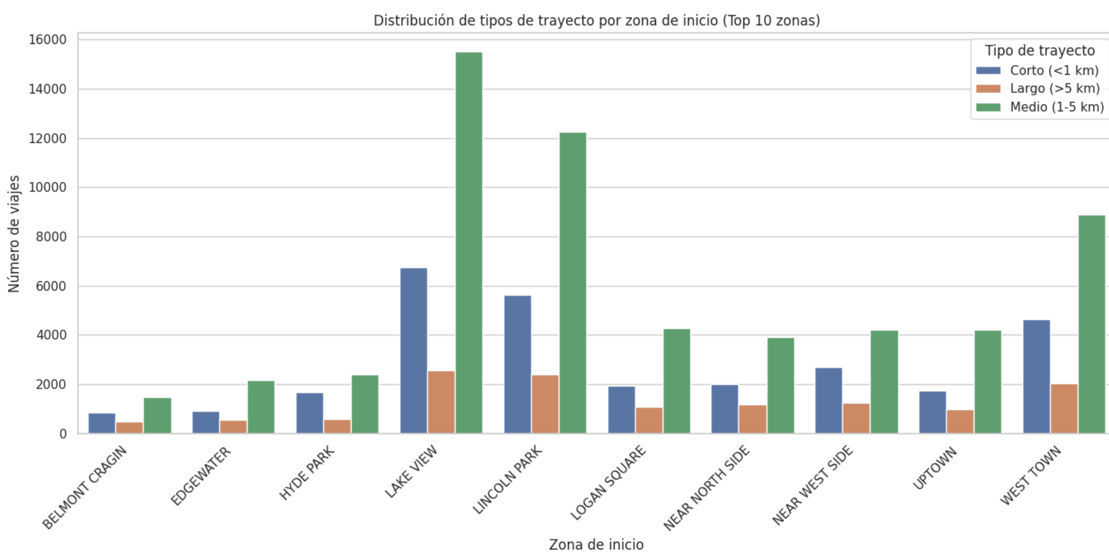
```
/tmp/ipython-input-28067412.py:21: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x=trayectos_tipo.index, y=trayectos_tipo.values, palette='Set2')
```



Medio (1-5km) mayoritario; corto segundo. Uso urbano.

[ ]:
```
df['Tipo de Trayecto'] = df['Trip Distance (km)'].apply(clasificar_trayecto)

# agrupamos por zona de inicio y tipo de trayecto
zona_tipo = df.groupby(['Start Community Area Name', 'Tipo de Trayecto']).
 ↪size().reset_index(name='Total Viajes')

# filtramos 10 zonas con más viajes totales
zonas_top = zona_tipo.groupby('Start Community Area Name')['Total Viajes'].
 ↪sum().nlargest(10).index
zona_tipo_top = zona_tipo[zona_tipo['Start Community Area Name'].
 ↪isin(zonas_top)]
```

```
plt.figure(figsize=(14, 7))
sns.barplot(data=zona_tipo_top, x='Start Community Area Name', y='Total␣
↪Viajes', hue='Tipo de Trayecto')
plt.title('Distribución de tipos de trayecto por zona de inicio (Top 10 zonas)')
plt.xlabel('Zona de inicio')
plt.ylabel('Número de viajes')
plt.xticks(rotation=45, ha='right')
plt.legend(title='Tipo de trayecto')
plt.tight_layout()
plt.show()
```

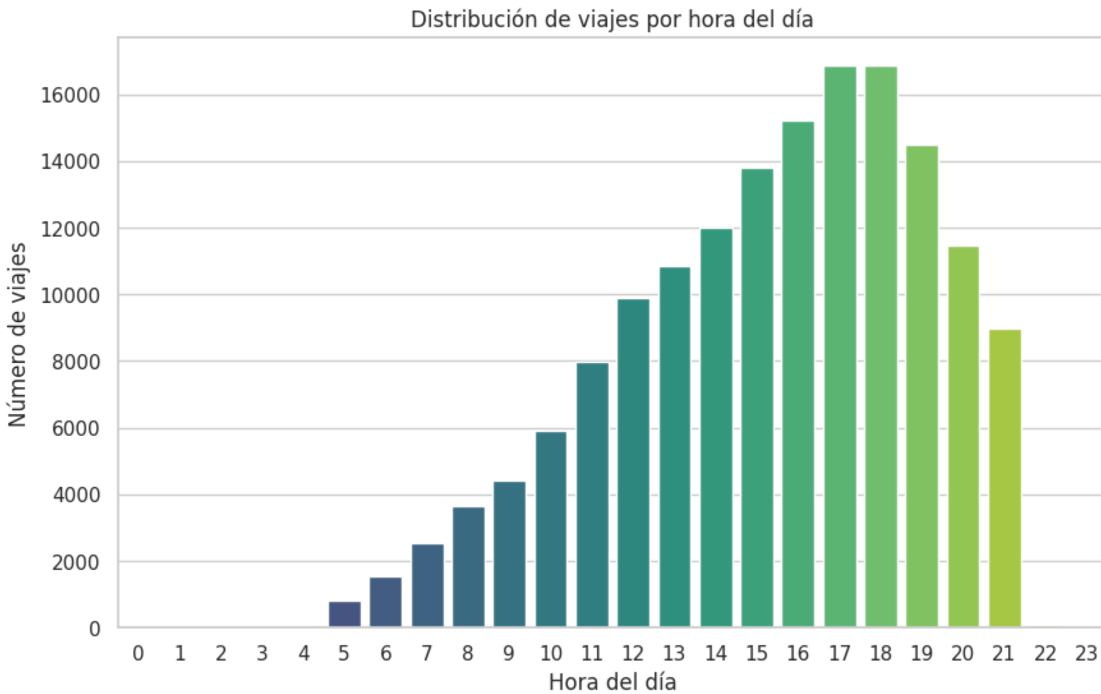


Medio domina en top zonas; patrones locales.

[ ]:
```
# Dsitribucion de viajes por hora del dia

plt.figure(figsize=(10,6))
sns.countplot(x='Hora_dia', data=df, palette='viridis')
plt.title('Distribución de viajes por hora del día')
plt.xlabel('Hora del día')
plt.ylabel('Número de viajes')
plt.show()
```

```
/tmp/ipython-input-3842621236.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.
```

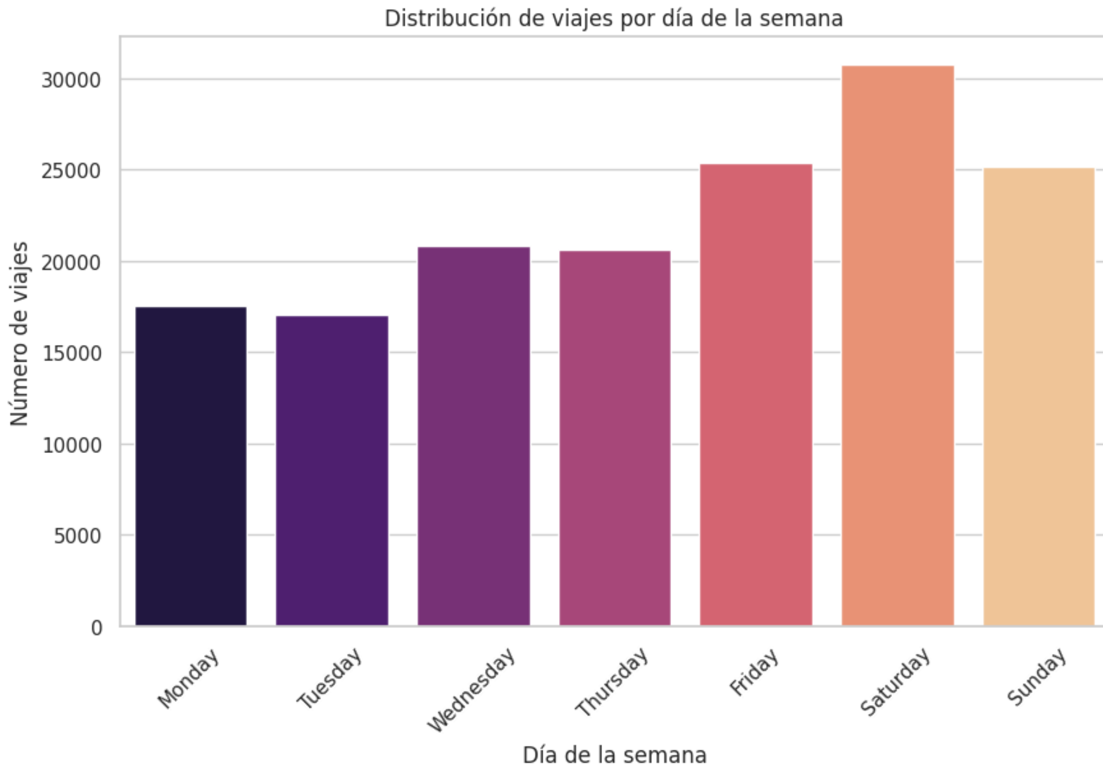
```
  sns.countplot(x='Hora_dia', data=df, palette='viridis')
```



Pico tarde; diurno.

```
[ ]: # Distribucion de viajes por dia de la semana

plt.figure(figsize=(10,6))
sns.countplot(x='Dia_semana', data=df, palette='magma', order=['Monday',␣
 ↪'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])
plt.title('Distribución de viajes por día de la semana')
plt.xlabel('Día de la semana')
plt.ylabel('Número de viajes')
plt.xticks(rotation=45)
plt.show()
```

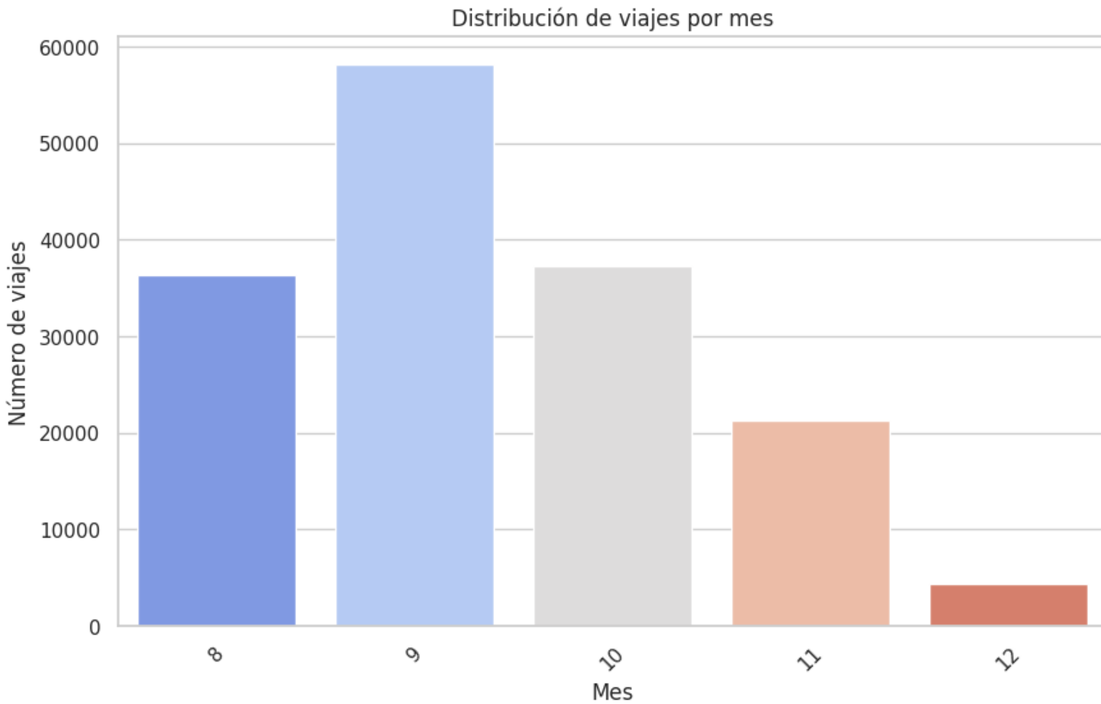
```
/tmp/ipython-input-1517760700.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Dia_semana', data=df, palette='magma', order=['Monday',
'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'])
```



Más fines de semana; recreativo.

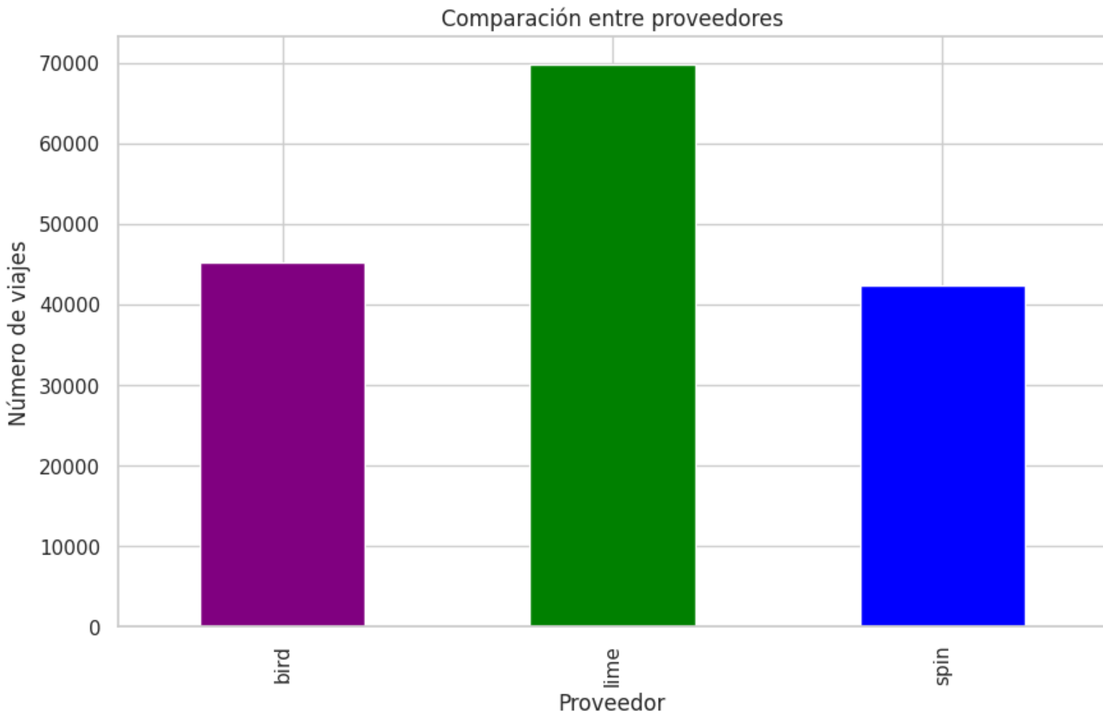
[ ]:
```
# Distribucion de viajes por mes

plt.figure(figsize=(10,6))
sns.countplot(x='Mes', data=df, palette='coolwarm')
plt.title('Distribución de viajes por mes')
plt.xlabel('Mes')
plt.ylabel('Número de viajes')
plt.xticks(rotation=45)
plt.show()
```

```
/tmp/ipython-input-495117003.py:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='Mes', data=df, palette='coolwarm')
```



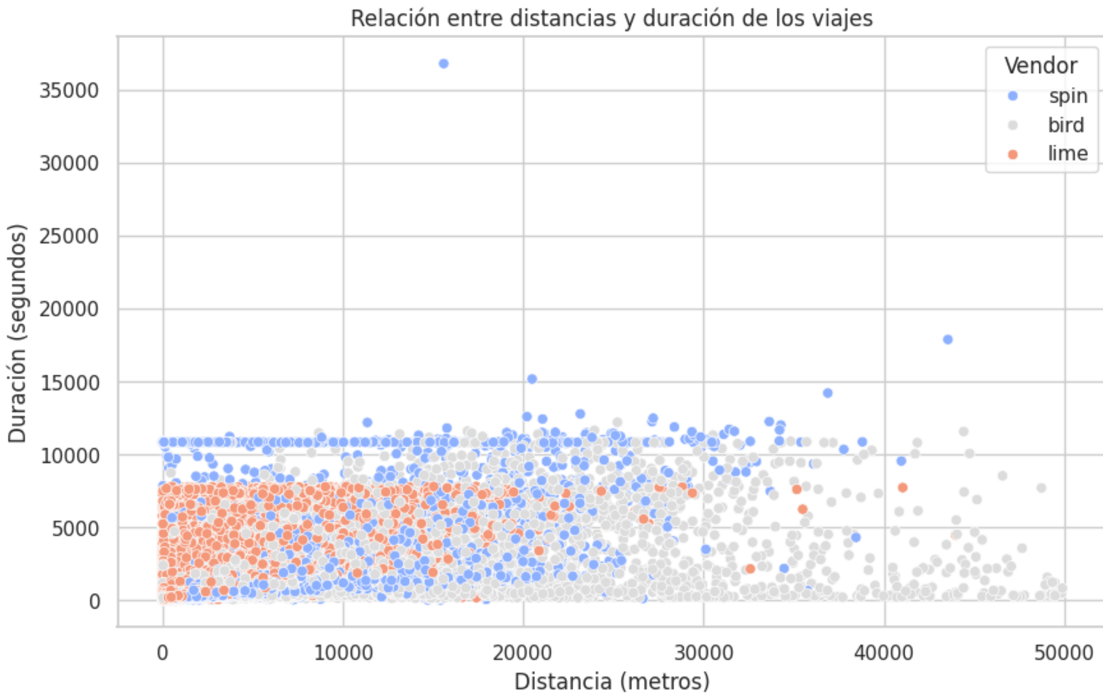
Pico verano; declive invierno (clima).

[ ]:
```
# Comparacion entre proveedores

plt.figure(figsize=(10,6))
df.groupby('Vendor').size().plot(kind='bar', color=['purple', 'green', 'blue'])
plt.title('Comparación entre proveedores')
plt.xlabel('Proveedor')
plt.ylabel('Número de viajes')
plt.show()
```



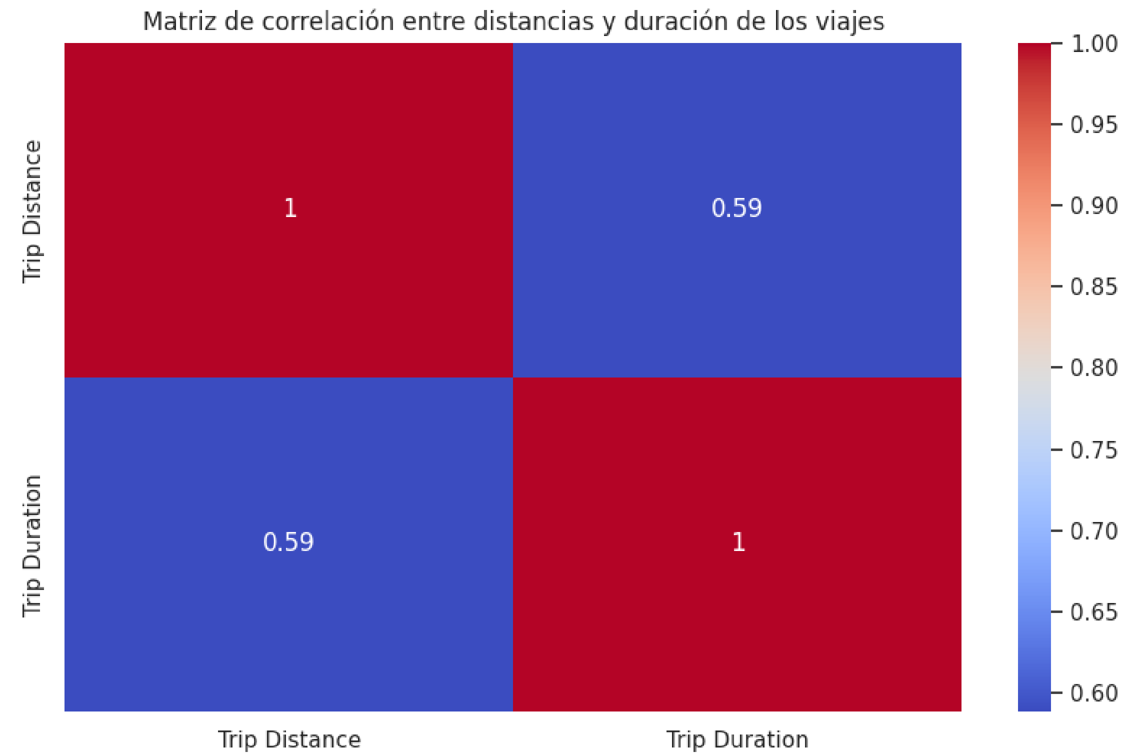
Desigualdad proveedores.

[ ]:
```
# relacion entre distancias y duracion

plt.figure(figsize=(10,6))
sns.scatterplot(x='Trip Distance', y='Trip Duration', data=df, hue='Vendor',␣
    ↪palette='coolwarm')
plt.title('Relación entre distancias y duración de los viajes')
plt.xlabel('Distancia (metros)')
plt.ylabel('Duración (segundos)')
plt.show()
```



Correlación positiva; clusters por proveedor.

[ ]:
```
# Matriz de correlacion

# Calculate the correlation matrix
corr_matrix = df[['Trip Distance', 'Trip Duration']].corr()

plt.figure(figsize=(10,6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Matriz de correlación entre distancias y duración de los viajes')
plt.show()
```



Distancia explica duración parcialmente.

[ ]:
```
# Analisis de uso por zonas

start_zones=df['Start Community Area Name'].value_counts().head(10)
end_zones=df['End Community Area Name'].value_counts().head(10)
print("\nTop 10 Zonas de inicio:")
print(start_zones)
print("\nTop 10 Zonas de fin:")
print(end_zones)
```

```
Top 10 Zonas de inicio:
Start Community Area Name
LAKE VIEW          24816
LINCOLN PARK       20246
WEST TOWN          15557
NEAR WEST SIDE      8147
LOGAN SQUARE        7299
NEAR NORTH SIDE     7123
UPTOWN              6930
HYDE PARK           4660
EDGEWATER           3656
```

```
BELMONT CRAGIN      2802
Name: count, dtype: int64

Top 10 Zonas de fin:
End Community Area Name
LAKE VIEW          24686
LINCOLN PARK       19818
WEST TOWN          15540
NEAR WEST SIDE      8120
LOGAN SQUARE        7382
NEAR NORTH SIDE     7271
UPTOWN              6768
HYDE PARK           4553
EDGEWATER           3711
BELMONT CRAGIN      2712
Name: count, dtype: int64
```

Hotspots centrales.

[ ]:
```
pip install pandas seaborn matplotlib folium
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(2.2.2)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-
packages (0.13.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-
packages (3.10.0)
Requirement already satisfied: folium in /usr/local/lib/python3.11/dist-packages
(0.20.0)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-
packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-
packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (4.59.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-
```

```
packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2.3)
Requirement already satisfied: branca>=0.6.0 in /usr/local/lib/python3.11/dist-
packages (from folium) (0.8.1)
Requirement already satisfied: jinja2>=2.9 in /usr/local/lib/python3.11/dist-
packages (from folium) (3.1.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from folium) (2.32.3)
Requirement already satisfied: xyzservices in /usr/local/lib/python3.11/dist-
packages (from folium) (2025.4.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from jinja2>=2.9->folium) (3.0.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->folium) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests->folium) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->folium) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->folium) (2025.8.3)
```

[ ]:
```python
# Mapa de calor de puntos de inicio
import folium
from folium.plugins import HeatMap

m_start=folium.Map(location=[df['Start Centroid Latitude'].mean(), df['Start
 ↪Centroid Longitude'].mean()], zoom_start=12)
HeatMap(data=df[['Start Centroid Latitude', 'Start Centroid Longitude']],
 ↪radius=10).add_to(m_start)
m_start
```

```
Output hidden; open in https://colab.research.google.com to view.
```

[ ]:
```python
# Mapa de calor de puntos de inicio
m_start = folium.Map(location=[41.8781, -87.6298], zoom_start=11)  # Centrado
 ↪en Chicago
heat_data_start = df[['Start Centroid Latitude', 'Start Centroid Longitude']].
 ↪values.tolist()
HeatMap(heat_data_start).add_to(m_start)
m_start
```

```
Output hidden; open in https://colab.research.google.com to view.
```

Densidad centro/norte Chicago.

[ ]:
```
# Mapa de calor de puntos de fin
m_end = folium.Map(location=[41.8781, -87.6298], zoom_start=11)
heat_data_end = df[['End Centroid Latitude', 'End Centroid Longitude']].values.
    ↪tolist()
HeatMap(heat_data_end).add_to(m_end)
m_end
```

Output hidden; open in https://colab.research.google.com to view.

Igual, hotspots urbanos.

Similar a start; muchos circulares.

[ ]:
```
# Seleccionar las 10 zonas con mayor actividad (inicio o fin)
top_zones = set(start_zones.head(10).index).union(set(end_zones.head(10).index))
flow_df = df[df['Start Community Area Name'].isin(top_zones) & df['End␣
    ↪Community Area Name'].isin(top_zones)]
flow_counts = flow_df.groupby(['Start Community Area Name', 'End Community Area␣
    ↪Name']).size().reset_index(name='count')
top_zones
```

[ ]:
```
{'BELMONT CRAGIN',
 'EDGEWATER',
 'HYDE PARK',
 'LAKE VIEW',
 'LINCOLN PARK',
 'LOGAN SQUARE',
 'NEAR NORTH SIDE',
 'NEAR WEST SIDE',
 'UPTOWN',
 'WEST TOWN'}
```

[ ]:
```
# Preparar datos para el diagrama de Sankey
labels = list(set(flow_counts['Start Community Area Name']).
    ↪union(set(flow_counts['End Community Area Name'])))
label_to_index = {label: idx for idx, label in enumerate(labels)}
sources = flow_counts['Start Community Area Name'].map(label_to_index)
targets = flow_counts['End Community Area Name'].map(label_to_index)
values = flow_counts['count']
labels
```

[ ]:
```
['EDGEWATER',
 'WEST TOWN',
 'HYDE PARK',
 'BELMONT CRAGIN',
 'NEAR NORTH SIDE',
 'LAKE VIEW',
 'UPTOWN',
```

```
'LOGAN SQUARE',
'LINCOLN PARK',
'NEAR WEST SIDE']
```

[ ]:
```
import plotly.graph_objects as go
```

[ ]:
```
fig=go.Figure(data=[go.Sankey(node=dict(label=labels),␣
 ↪link=dict(source=sources, target=targets, value=values))])
fig.update_layout(title_text='Flujo entre Zonas de Inicio y Fin', font_size=10,␣
 ↪height=1800)
fig.write_html('sankey_flows.html')
fig.show()
```

Flujos intra-zona fuertes; inter adyacentes.

[ ]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.ensemble import RandomForestRegressor
```

[ ]:
```
# codificar las variables categoricas

le_vendor=LabelEncoder()
le_dia=LabelEncoder()
le_mes=LabelEncoder()
le_start_zone=LabelEncoder()
le_end_zone=LabelEncoder()
```

[ ]:
```
df['Vendor_Enconder']=le_vendor.fit_transform(df['Vendor'])
df['Dia_semana_Encoded']=le_dia.fit_transform(df['Dia_semana'])
df['Dia_mes_Encoded']=le_dia.fit_transform(df['Mes'])
df['Start_zone_Encoded']=le_start_zone.fit_transform(df['Start Community Area␣
 ↪Name'])
df['End_zone_Encoded']=le_end_zone.fit_transform(df['End Community Area Name'])
```

[ ]:
```
features=['Trip Distance', 'Vendor_Enconder', 'Hora_dia', 'Dia_semana_Encoded',␣
 ↪'Dia_mes_Encoded', 'Start_zone_Encoded', 'End_zone_Encoded', ]
X=df[features]
y=df['Trip Duration']
```

[ ]:
```
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[ ]:
```
rf_model=RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

[ ]: RandomForestRegressor(random_state=42)

```python
y_pred=rf_model.predict(X_test)
mse=mean_squared_error(y_test, y_pred)
r2=r2_score(y_test, y_pred)
print("\nResultados del Modelo Random Forest:")
print(f"Error Cuadratico Medio (MSE): {mse:2f}")
print(f"Coeficiente de Determinacion (R^2): {r2:2f}")
```

```
Resultados del Modelo Random Forest:
Error Cuadratico Medio (MSE): 944685.262172
Coeficiente de Determinacion (R^2): 0.429560
```

```python
feature_importance=pd.DataFrame({'feature': features, 'Importance': rf_model.
    ↪feature_importances_})
feature_importance=feature_importance.sort_values('Importance', ascending=False)
print("\nImportancia de las caracteristicas:")
print(feature_importance)
```

```
Importancia de las caracteristicas:
              feature  Importance
0       Trip Distance    0.594969
6     End_zone_Encoded    0.092476
5   Start_zone_Encoded    0.085248
2             Hora_dia    0.083084
3   Dia_semana_Encoded    0.060177
4      Dia_mes_Encoded    0.048422
1      Vendor_Enconder    0.035623
```

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler, label_binarize
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.metrics import classification_report, confusion_matrix,␣
    ↪ConfusionMatrixDisplay, roc_curve, auc
```

```python
top_areas = df['End Community Area Name'].value_counts().nlargest(10).index
df = df[df['End Community Area Name'].isin(top_areas)]


X = df[['Hour', 'Trip Distance', 'Trip Duration', 'Vendor',
        'Start Community Area Name', 'Tipo de Trayecto']]
y = df['End Community Area Name']
```

```python
# columnas numéricas y categóricas
numeric_features = ['Hour', 'Trip Distance', 'Trip Duration']
categorical_features = ['Vendor', 'Start Community Area Name', 'Tipo de␣
 ↪Trayecto']

# Preprocesadores
numeric_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

preprocessor = ColumnTransformer([
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])

# Pipeline con regresión logística
clf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(multi_class='multinomial', max_iter=1000))
])


X_train, X_test, y_train, y_test = train_test_split(
    X, y, stratify=y, test_size=0.3, random_state=42)


clf.fit(X_train, y_train)


y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

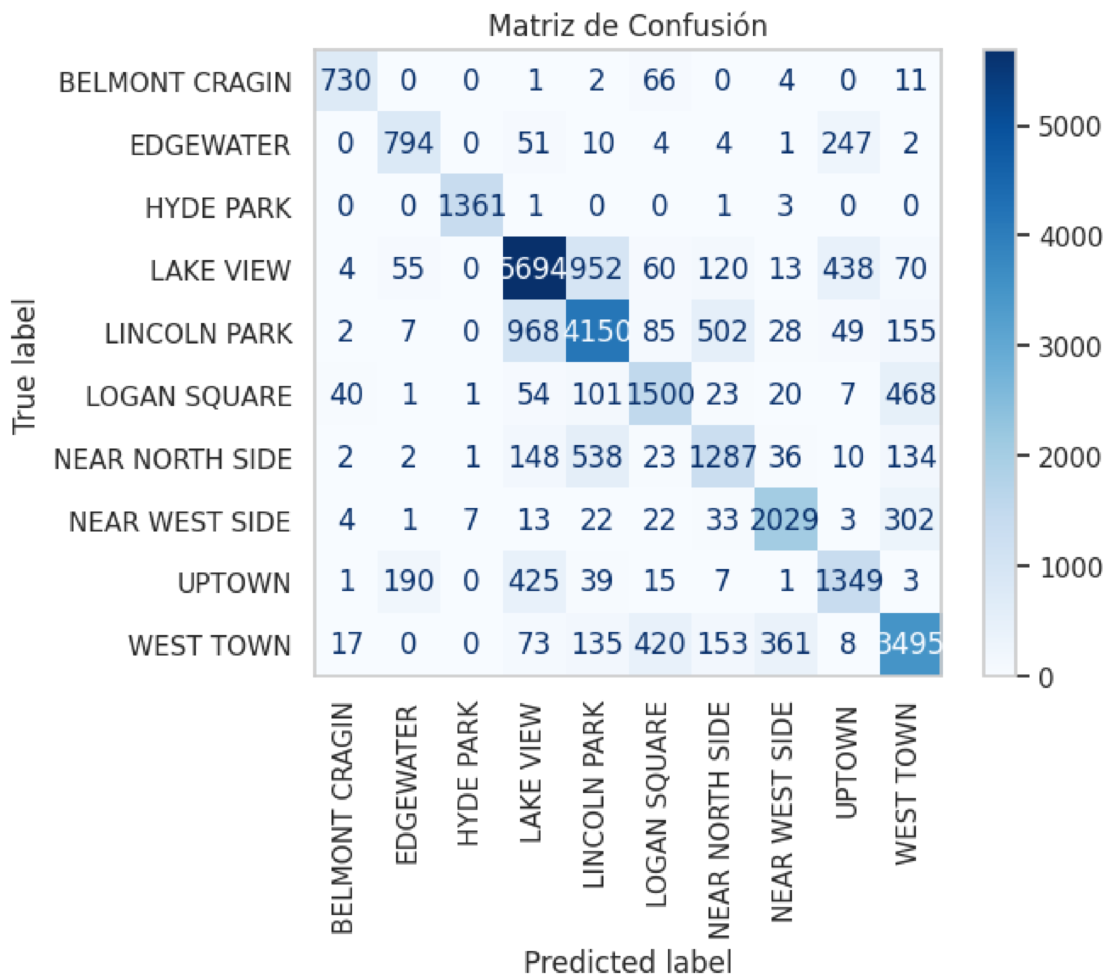
```
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247:
FutureWarning:

'multi_class' was deprecated in version 1.5 and will be removed in 1.7. From
then on, it will always use 'multinomial'. Leave it to its default value to
avoid this warning.


                 precision    recall  f1-score   support

 BELMONT CRAGIN       0.91      0.90      0.90       814
      EDGEWATER       0.76      0.71      0.73      1113
      HYDE PARK       0.99      1.00      0.99      1366
      LAKE VIEW       0.77      0.77      0.77      7406
   LINCOLN PARK       0.70      0.70      0.70      5946
   LOGAN SQUARE       0.68      0.68      0.68      2215
NEAR NORTH SIDE       0.60      0.59      0.60      2181
```

```
NEAR WEST SIDE       0.81      0.83      0.82      2436
        UPTOWN       0.64      0.66      0.65      2030
     WEST TOWN       0.75      0.75      0.75      4662

      accuracy                           0.74     30169
     macro avg       0.76      0.76      0.76     30169
  weighted avg       0.74      0.74      0.74     30169
```

Bueno en zonas populares.

[ ]:
```
cm = confusion_matrix(y_test, y_pred, labels=clf.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.classes_)

plt.figure(figsize=(16, 12))
disp.plot(xticks_rotation=90, cmap='Blues')
plt.title("Matriz de Confusión")
plt.grid(False)
plt.show()
```

```
<Figure size 1600x1200 with 0 Axes>
```

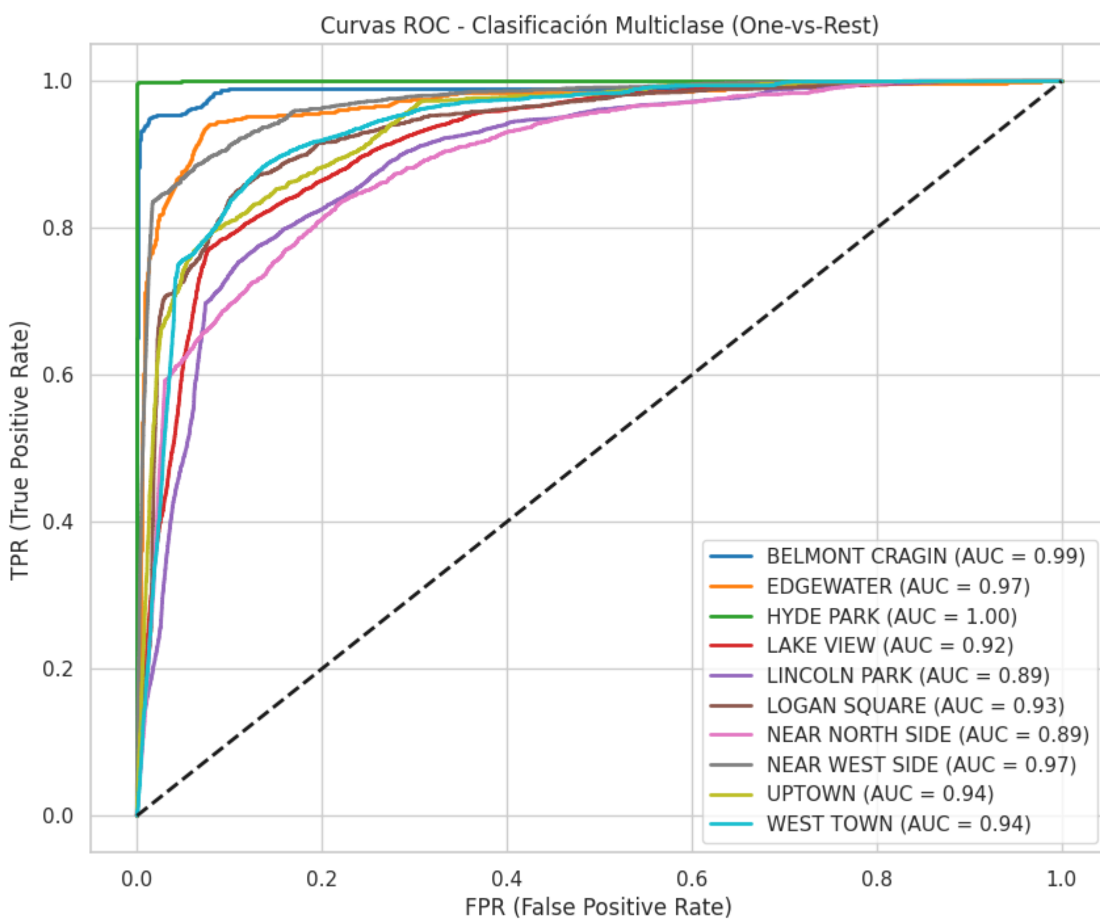


Confusiones entre adyacentes.

```
[ ]: # binarizar las etiquetas
y_test_bin = label_binarize(y_test, classes=clf.classes_)
n_classes = y_test_bin.shape[1]

# predecir probabilidades
y_score = clf.predict_proba(X_test)

# curvas ROC
fpr, tpr, roc_auc = {}, {}, {}
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test_bin[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

plt.figure(figsize=(10, 8))
colors = plt.cm.get_cmap('tab10', n_classes)
```

```python
for i in range(n_classes):
    plt.plot(fpr[i], tpr[i], label=f'{clf.classes_[i]} (AUC = {roc_auc[i]:.
 ↪2f})', lw=2, color=colors(i))
plt.plot([0, 1], [0, 1], 'k--', lw=2)
plt.xlabel('FPR (False Positive Rate)')
plt.ylabel('TPR (True Positive Rate)')
plt.title('Curvas ROC - Clasificación Multiclase (One-vs-Rest)')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```

```
/tmp/ipython-input-2867315746.py:15: MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be removed in
3.11. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap()``
or ``pyplot.get_cmap()`` instead.
```



```
[ ]: from sklearn.ensemble import RandomForestClassifier
```

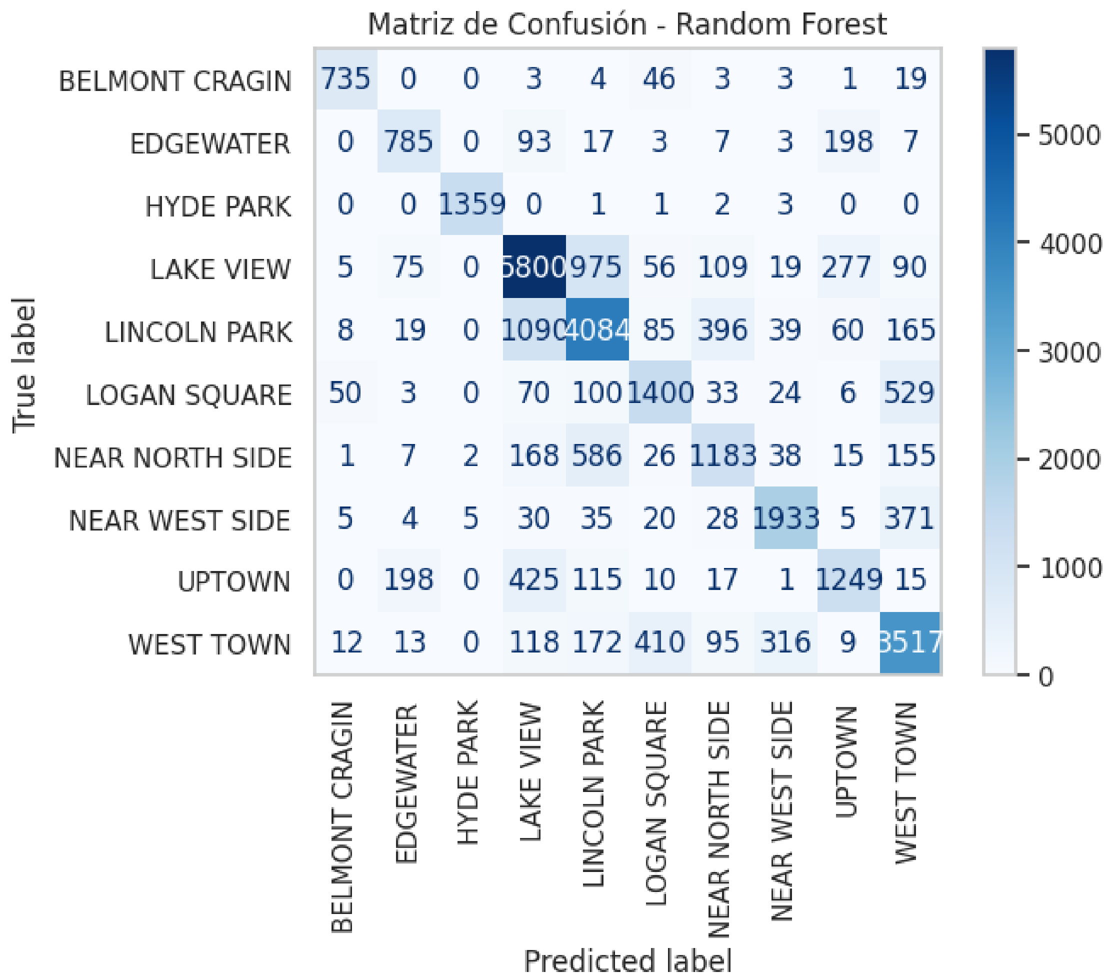
```
[ ]: # Pipeline con Random Forest
clf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])


X_train, X_test, y_train, y_test = train_test_split(
    X, y, stratify=y, test_size=0.3, random_state=42)


clf.fit(X_train, y_train)


y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
                 precision    recall  f1-score   support

 BELMONT CRAGIN       0.90      0.90      0.90       814
      EDGEWATER       0.71      0.71      0.71      1113
      HYDE PARK       0.99      0.99      0.99      1366
      LAKE VIEW       0.74      0.78      0.76      7406
   LINCOLN PARK       0.67      0.69      0.68      5946
   LOGAN SQUARE       0.68      0.63      0.66      2215
NEAR NORTH SIDE       0.63      0.54      0.58      2181
 NEAR WEST SIDE       0.81      0.79      0.80      2436
         UPTOWN       0.69      0.62      0.65      2030
      WEST TOWN       0.72      0.75      0.74      4662

       accuracy                           0.73     30169
      macro avg       0.76      0.74      0.75     30169
   weighted avg       0.73      0.73      0.73     30169
```

Menos confusiones top zones.

```
[ ]: cm = confusion_matrix(y_test, y_pred, labels=clf.named_steps['classifier'].
  ↪classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=clf.
  ↪named_steps['classifier'].classes_)
plt.figure(figsize=(10, 8))
disp.plot(xticks_rotation=90, cmap='Blues')
plt.title("Matriz de Confusión - Random Forest")
plt.grid(False)
plt.show()
```

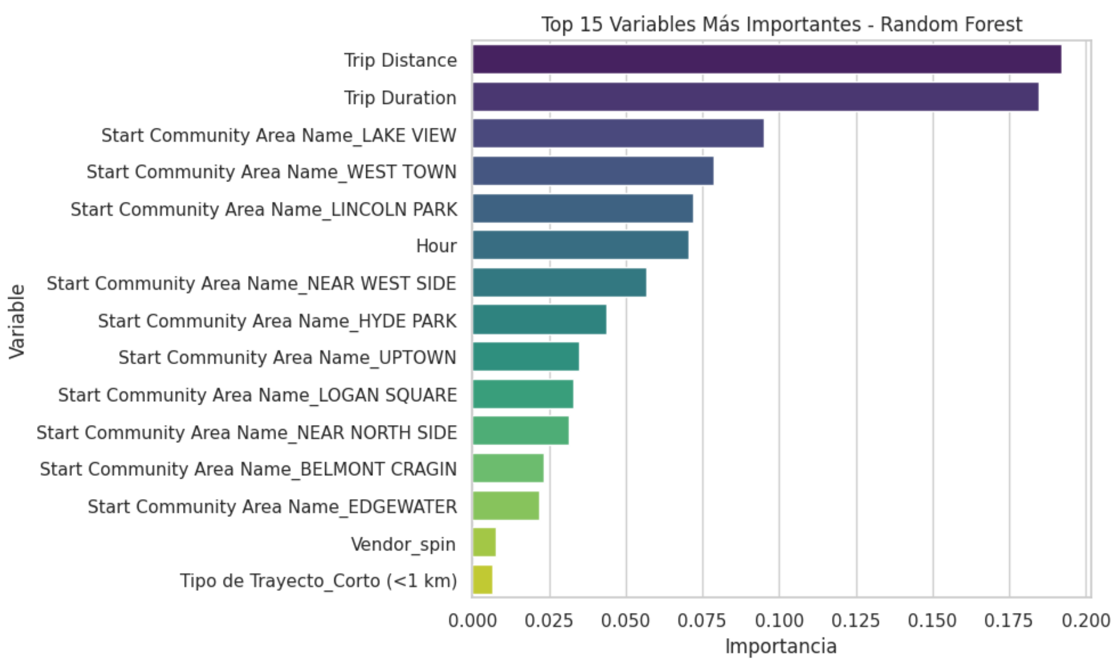
<Figure size 1000x800 with 0 Axes>




```
[ ]: rf_model = clf.named_steps['classifier']

ohe = clf.named_steps['preprocessor'].named_transformers_['cat']
cat_columns = ohe.get_feature_names_out(categorical_features)
all_features = numeric_features + list(cat_columns)


importancias = rf_model.feature_importances_


importancia_df = pd.DataFrame({
    'Feature': all_features,
    'Importance': importancias
}).sort_values(by='Importance', ascending=False).head(15)
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importancia_df, palette='viridis')
plt.title('Top 15 Variables Más Importantes - Random Forest')
plt.xlabel('Importancia')
plt.ylabel('Variable')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-3940263193.py:18: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.
```



Duración/distancia clave.

```
[ ]: df['TrayectoCircular'] = df['Start Community Area Name'] == df['End Community␣
    ↪Area Name']
```

```
/tmp/ipython-input-3591060111.py:1: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

[ ]:
```
conteo_circulares = df['TrayectoCircular'].value_counts()
print(conteo_circulares)

# %
porcentaje_circulares = df['TrayectoCircular'].mean() * 100
print(f"Porcentaje de trayectos circulares: {porcentaje_circulares:.2f}%")
```

```
TrayectoCircular
True     71166
False    29395
Name: count, dtype: int64
Porcentaje de trayectos circulares: 70.77%
```

Mayoría local.

[ ]:
```
# ¿Desde qué zonas se realizan más trayectos circulares?
circular_por_zona = df[df['TrayectoCircular']].groupby('Start Community Area␣
↪Name').size().sort_values(ascending=False)
print(circular_por_zona.head(10))
```

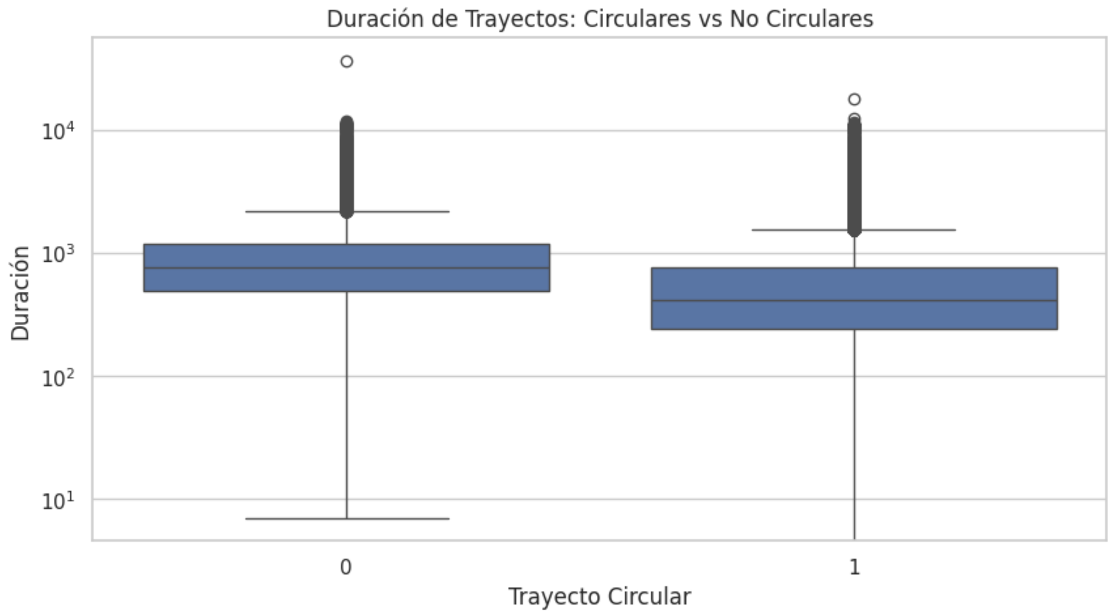
```
Start Community Area Name
LAKE VIEW          18399
LINCOLN PARK       13877
WEST TOWN          11378
NEAR WEST SIDE      6280
LOGAN SQUARE        4446
UPTOWN              4314
NEAR NORTH SIDE     4260
HYDE PARK           3836
EDGEWATER           2322
BELMONT CRAGIN      2054
dtype: int64
```

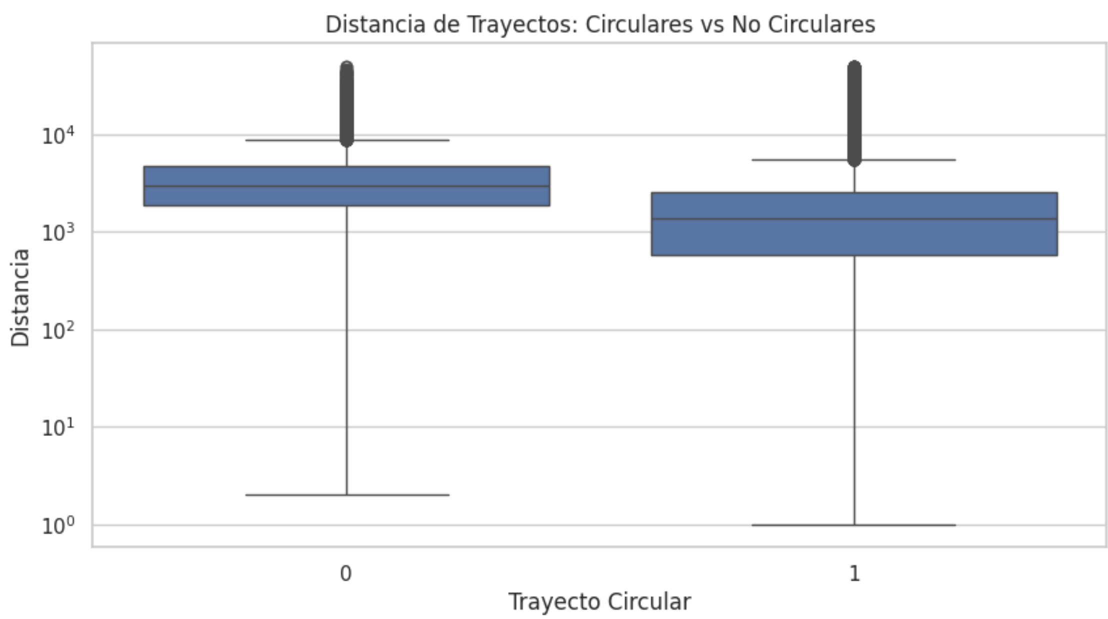
Zonas centrales más.

[ ]:
```
df = df.dropna(subset=['Start Community Area Name', 'End Community Area Name'])
df['TrayectoCircular'] = (df['Start Community Area Name'] == df['End Community␣
↪Area Name']).astype(int)

# variables predictoras
X = df[['Hour', 'Trip Distance', 'Trip Duration', 'Vendor',
```

```
        'Start Community Area Name', 'Tipo de Trayecto']]
y = df['TrayectoCircular']  # 1 si es circular, 0 si no


numeric_features = ['Hour', 'Trip Distance', 'Trip Duration']
categorical_features = ['Vendor', 'Start Community Area Name', 'Tipo de␣
↪Trayecto']

# preprocesamiento
numeric_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

preprocessor = ColumnTransformer([
    ('num', numeric_transformer, numeric_features),
    ('cat', categorical_transformer, categorical_features)
])


X_train, X_test, y_train, y_test = train_test_split(
    X, y, stratify=y, test_size=0.3, random_state=42)
```

[ ]:
```
plt.figure(figsize=(10, 5))
sns.boxplot(x='TrayectoCircular', y='Trip Duration', data=df)
plt.title("Duración de Trayectos: Circulares vs No Circulares")
plt.xlabel("Trayecto Circular")
plt.ylabel("Duración")
plt.yscale('log')  # Si hay valores atípicos grandes
plt.show()

plt.figure(figsize=(10, 5))
sns.boxplot(x='TrayectoCircular', y='Trip Distance', data=df)
plt.title("Distancia de Trayectos: Circulares vs No Circulares")
plt.xlabel("Trayecto Circular")
plt.ylabel("Distancia")
plt.yscale('log')
plt.show()
```





Circulares más cortos/rápidos.

[ ]:
```
logreg_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000))
])
```

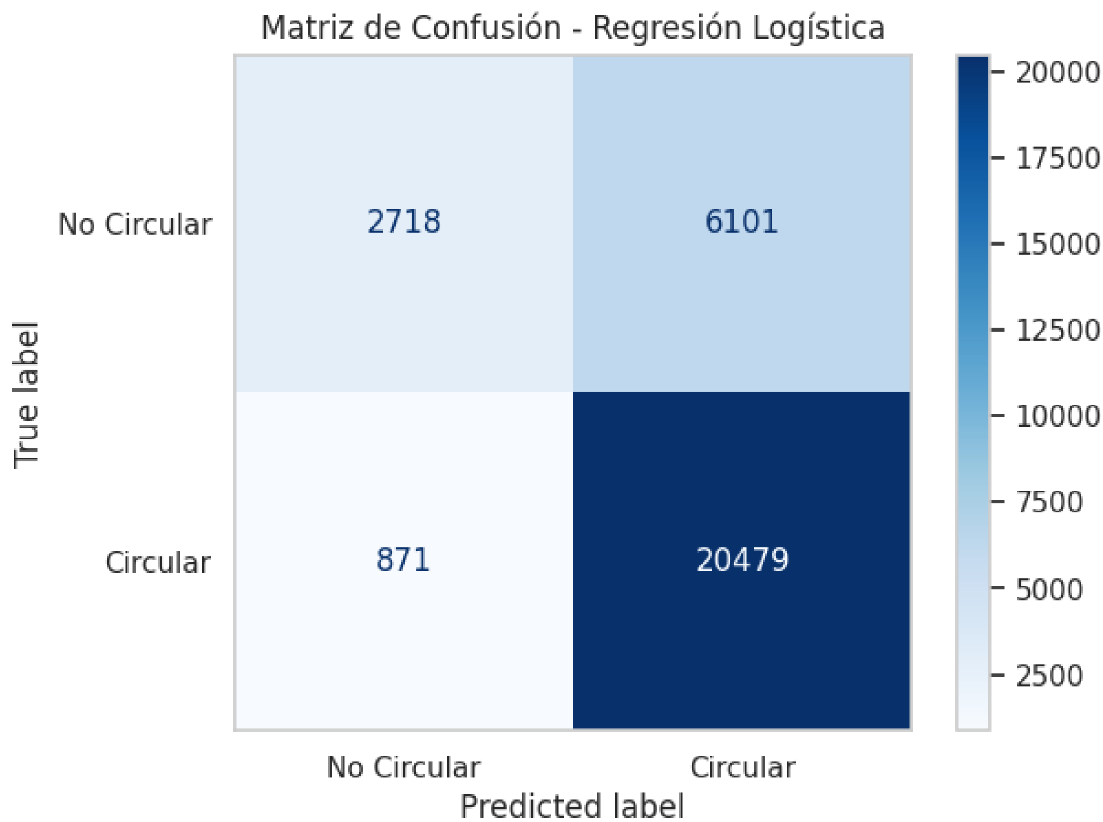
```
logreg_pipeline.fit(X_train, y_train)

print("Regresión Logística")
y_pred_lr = logreg_pipeline.predict(X_test)
print(classification_report(y_test, y_pred_lr))
```

```
Regresión Logística
              precision    recall  f1-score   support

           0       0.76      0.31      0.44      8819
           1       0.77      0.96      0.85     21350

    accuracy                           0.77     30169
   macro avg       0.76      0.63      0.65     30169
weighted avg       0.77      0.77      0.73     30169
```

Falsos positivos no-circulares.

[ ]:
```
labels = [0, 1]  # 0: No Circular, 1: Circular

cm_lr = confusion_matrix(y_test, y_pred_lr, labels=labels)
disp_lr = ConfusionMatrixDisplay(confusion_matrix=cm_lr, display_labels=['No␣
↪Circular', 'Circular'])

plt.figure(figsize=(6, 5))
disp_lr.plot(cmap='Blues', values_format='d')
plt.title("Matriz de Confusión - Regresión Logística")
plt.grid(False)
plt.show()
```

```
<Figure size 600x500 with 0 Axes>
```

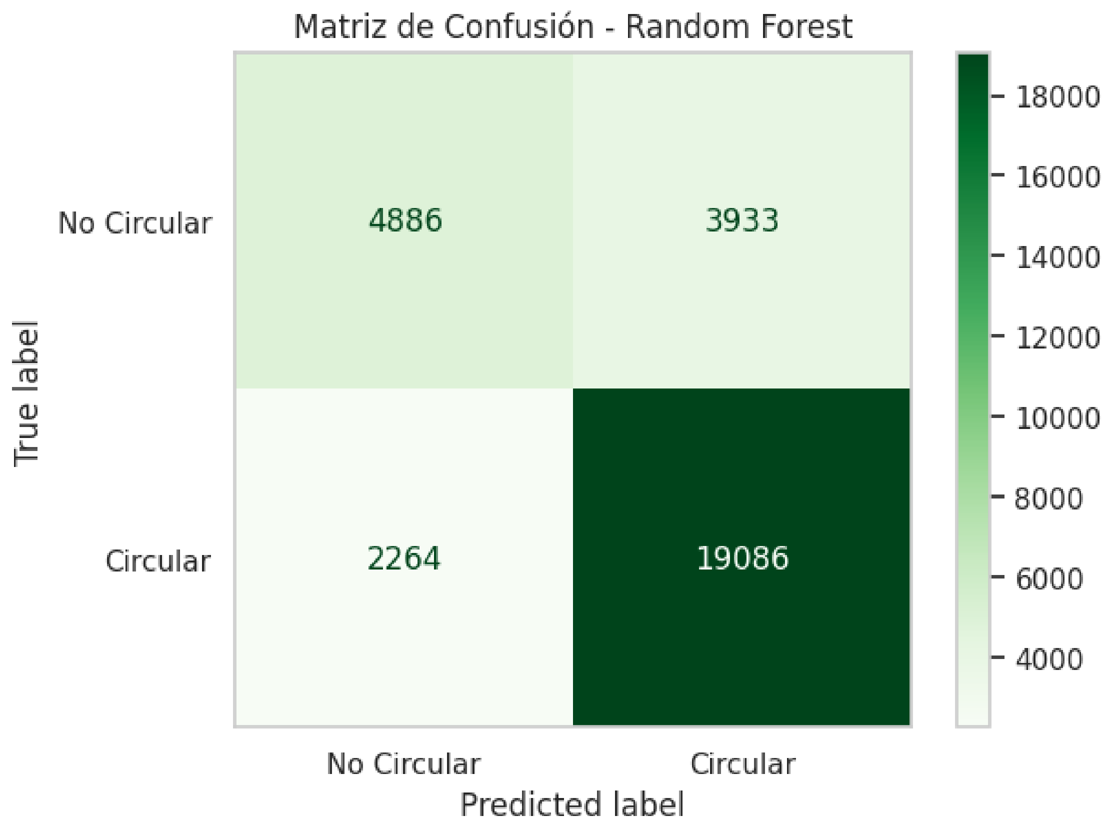


[ ]:
```
rf_pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(n_estimators=100, random_state=42))
])

rf_pipeline.fit(X_train, y_train)

print("Random Forest")
y_pred_rf = rf_pipeline.predict(X_test)
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest
              precision    recall  f1-score   support

           0       0.68      0.55      0.61      8819
           1       0.83      0.89      0.86     21350

    accuracy                           0.79     30169
   macro avg       0.76      0.72      0.74     30169
weighted avg       0.79      0.79      0.79     30169
```

[ ]:
```
cm_rf = confusion_matrix(y_test, y_pred_rf, labels=labels)
disp_rf = ConfusionMatrixDisplay(confusion_matrix=cm_rf, display_labels=['No␣
↪Circular', 'Circular'])

plt.figure(figsize=(6, 5))
disp_rf.plot(cmap='Greens', values_format='d')
plt.title("Matriz de Confusión - Random Forest")
plt.grid(False)
plt.show()
```

<Figure size 600x500 with 0 Axes>

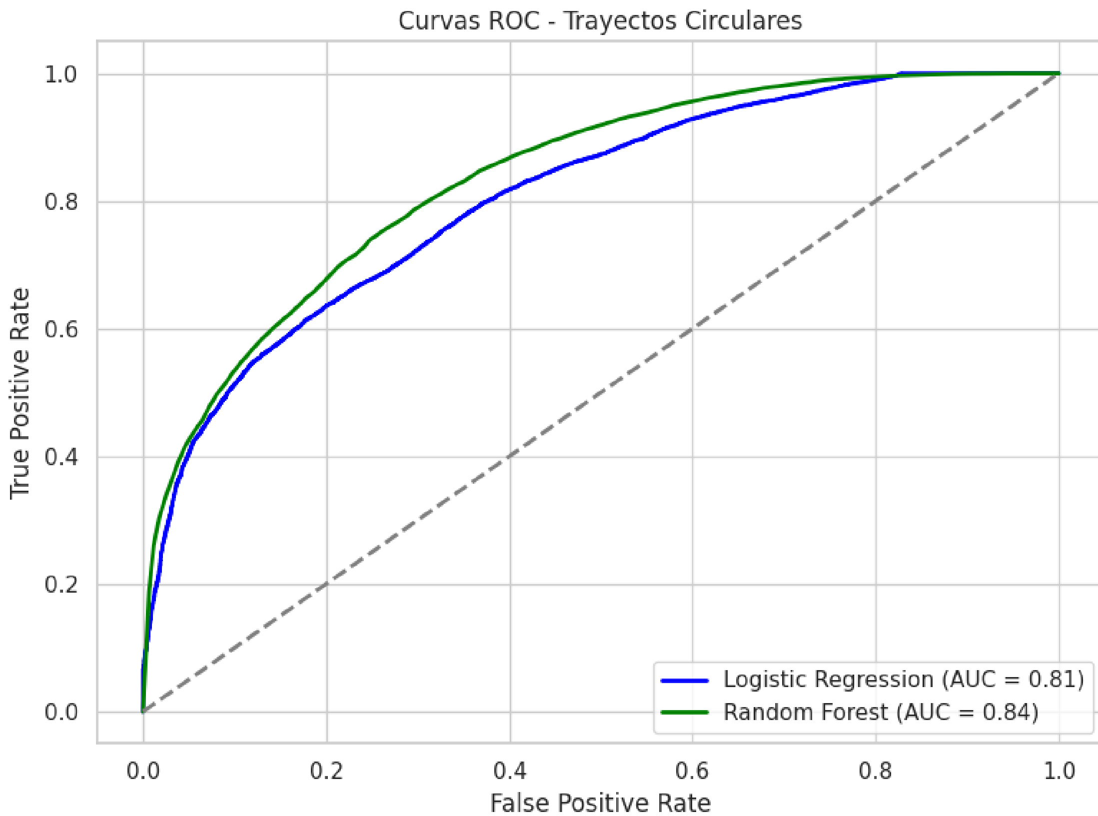


Calcula ROC y AUC para Logistic Regression (LR) y Random Forest (RF) en la predicción de trayectos circulares; plotea ambas curvas en un gráfico.

Uso roc_curve y auc de scikit-learn para evaluar el rendimiento discriminativo de ambos modelos en la tarea binaria (circular vs no circular). Plotea las curvas con una línea diagonal de referencia (clasificador aleatorio)

[ ]:
```
from sklearn.metrics import roc_curve, auc
```

```python
# probabilidades predichas (clase positiva = 1)
y_score_lr = logreg_pipeline.predict_proba(X_test)[:, 1]
y_score_rf = rf_pipeline.predict_proba(X_test)[:, 1]

# curvas ROC
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_score_lr)
roc_auc_lr = auc(fpr_lr, tpr_lr)

fpr_rf, tpr_rf, _ = roc_curve(y_test, y_score_rf)
roc_auc_rf = auc(fpr_rf, tpr_rf)


plt.figure(figsize=(8, 6))
plt.plot(fpr_lr, tpr_lr, color='blue', lw=2, label=f'Logistic Regression (AUC =␣
↪{roc_auc_lr:.2f})')
plt.plot(fpr_rf, tpr_rf, color='green', lw=2, label=f'Random Forest (AUC =␣
↪{roc_auc_rf:.2f})')
plt.plot([0, 1], [0, 1], color='gray', linestyle='--', lw=2)

plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Curvas ROC - Trayectos Circulares')
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
plt.show()
```



Ambos modelos superan al azar (AUC >0.5), con RF ligeramente mejor (mayor área bajo la curva, indicando mejor trade-off entre TPR y FPR). Sugiere que RF captura mejor las no-linealidades en features como distancia y hora para predecir si un viaje es circular. Buen rendimiento overall, pero espacio para mejora (AUC no llega a 0.9+).

Agrupa por 'Start Community Area Name' y 'End Community Area Name'; cuenta viajes; top 15 descendente.

Identifica los flujos más comunes entre áreas (OD pairs) para entender movilidad.

[ ]:
```
top_od = (df.groupby(['Start Community Area Name','End Community Area Name'])
            .size().sort_values(ascending=False)
            .head(15).reset_index(name='n'))

top_od
```

[ ]:
```
  Start Community Area Name End Community Area Name      n
0                 LAKE VIEW               LAKE VIEW  18399
1              LINCOLN PARK            LINCOLN PARK  13877
2                 WEST TOWN               WEST TOWN  11378
3            NEAR WEST SIDE          NEAR WEST SIDE   6280
4              LOGAN SQUARE            LOGAN SQUARE   4446
```

```
5                  UPTOWN                 UPTOWN   4314
6         NEAR NORTH SIDE        NEAR NORTH SIDE   4260
7               HYDE PARK              HYDE PARK   3836
8            LINCOLN PARK              LAKE VIEW   3278
9               LAKE VIEW           LINCOLN PARK   3168
10              EDGEWATER              EDGEWATER   2322
11         BELMONT CRAGIN         BELMONT CRAGIN   2054
12           LINCOLN PARK        NEAR NORTH SIDE   1793
13        NEAR NORTH SIDE           LINCOLN PARK   1642
14              WEST TOWN           LOGAN SQUARE   1412
```

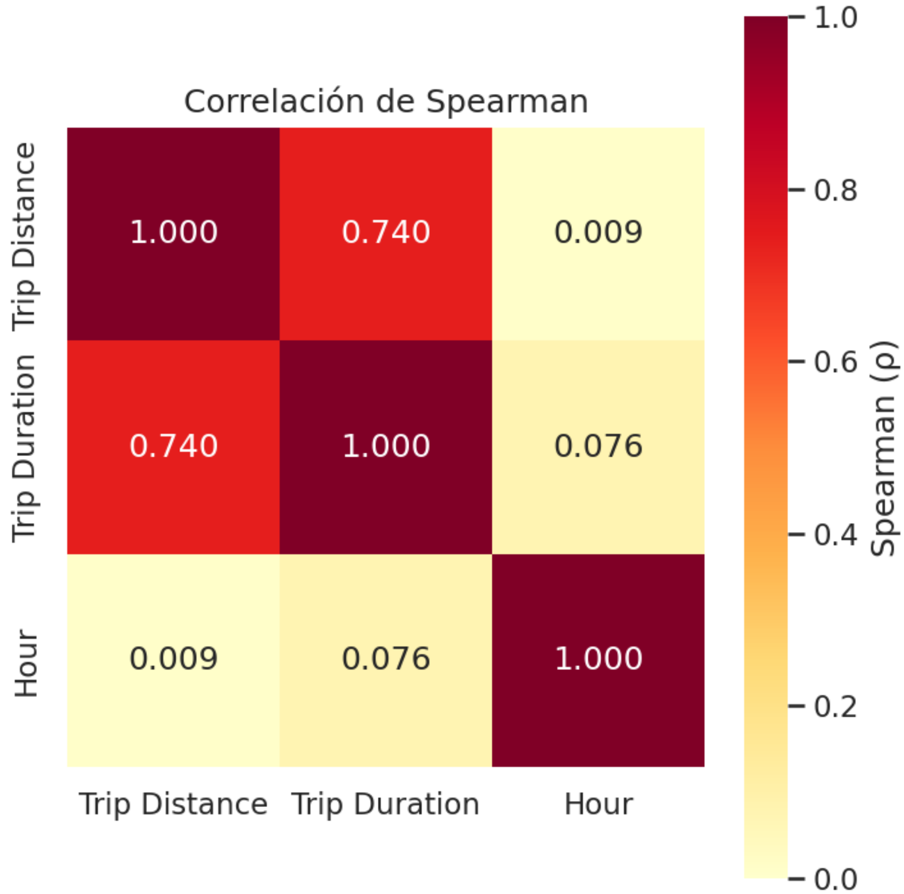
Dominan trayectos circulares en áreas centrales (e.g., Lake View, Lincoln Park). Inter-zonas como Lincoln Park Lake View indican conectividad entre vecindarios adyacentes. Refuerza que ~70% de viajes son locales.

[ ]:
```
df.columns
```

[ ]:
```
Index(['Date', 'Hour', 'Trip Distance', 'Trip Duration', 'Vendor',
       'Start Community Area Number', 'End Community Area Number',
       'Start Community Area Name', 'End Community Area Name',
       'Start Centroid Latitude', 'Start Centroid Longitude',
       'End Centroid Latitude', 'End Centroid Longitude', 'Dia_hora',
       'Dia_semana', 'Mes', 'Hora_dia', 'Trip Distance (km)',
       'Tipo de Trayecto', 'Vendor_Enconder', 'Dia_semana_Encoded',
       'Dia_mes_Encoded', 'Start_zone_Encoded', 'End_zone_Encoded',
       'TrayectoCircular'],
      dtype='object')
```

Calculo correlación Spearman entre 'Trip Distance', 'Trip Duration' y 'Hour'; heatmap con Seaborn.

[ ]:
```
cols = ['Trip Distance', 'Trip Duration', 'Hour']
corr_spear = df[cols].corr(method='spearman')

plt.figure(figsize=(5, 5),dpi=150)
sns.heatmap(corr_spear, annot=True, fmt=".3f", vmin=0, vmax=1,
            cmap='YlOrRd', square=True, cbar_kws={'label': 'Spearman ( )'})
plt.title('Correlación de Spearman')
plt.tight_layout()
plt.show()
```



Fuerte correlación positiva entre distancia y duración (mayor distancia implica más tiempo, como esperado). Correlaciones débiles con hora (uso no varía mucho por hora en términos monotónicos). Spearman confirma robustez vs outliers.

[ ]:
```
df['Trip Distance (km)'] = df['Trip Distance'] / 1000 # convertimos a kilometros


def clasificar_trayecto(distancia):
    if distancia < 1:
        return 'Corto (<1 km)'
    elif distancia <= 5:
        return 'Medio (1-5 km)'
    else:
        return 'Largo (>5 km)'


df['Tipo de Trayecto'] = df['Trip Distance (km)'].apply(clasificar_trayecto)
```

```
df
```

```
[ ]:                Date  Hour  Trip Distance  Trip Duration Vendor  \
1        08/12/2020     7             13            101   spin
2        08/12/2020     7              7             50   bird
3        08/12/2020     7           3815            840   spin
4        08/12/2020     8           1444            445   spin
5        08/12/2020     8             15            110   spin
…               …    …            …             …     …
157284   12/12/2020    20           3005            535   spin
157287   12/12/2020    21           3410            683   lime
157291   12/12/2020    21           9257           2214   spin
157292   12/12/2020    21            878            325   lime
157293   12/12/2020    21            490            212   lime

        Start Community Area Number  End Community Area Number  \
1                               7.0                        7.0
2                              77.0                       77.0
3                               6.0                        3.0
4                               3.0                        6.0
5                               6.0                        6.0
…                                …                         …
157284                          7.0                        7.0
157287                         23.0                       24.0
157291                          6.0                        6.0
157292                         28.0                       24.0
157293                          8.0                        8.0

       Start Community Area Name End Community Area Name  \
1                   LINCOLN PARK            LINCOLN PARK
2                      EDGEWATER               EDGEWATER
3                      LAKE VIEW                  UPTOWN
4                         UPTOWN               LAKE VIEW
5                      LAKE VIEW               LAKE VIEW
…                            …                     …
157284              LINCOLN PARK            LINCOLN PARK
157287             HUMBOLDT PARK               WEST TOWN
157291                 LAKE VIEW               LAKE VIEW
157292            NEAR WEST SIDE               WEST TOWN
157293           NEAR NORTH SIDE         NEAR NORTH SIDE

        Start Centroid Latitude  …  Mes  Hora_dia  Trip Distance (km)  \
1                     41.921880  …    8         7               0.013
2                     41.987114  …    8         7               0.007
3                     41.943514  …    8         7               3.815
4                     41.965435  …    8         8               1.444
5                     41.943514  …    8         8               0.015
```

```
…                         …   …   …       …                 …
157284                41.921880  …   12         20                3.005
157287                41.900813  …   12         21                3.410
157291                41.943514  …   12         21                9.257
157292                41.874254  …   12         21                0.878
157293                41.899528  …   12         21                0.490

       Tipo de Trayecto Vendor_Enconder  Dia_semana_Encoded  Dia_mes_Encoded  \
1         Corto (<1 km)               2                   6                0
2         Corto (<1 km)               0                   6                0
3        Medio (1-5 km)               2                   6                0
4        Medio (1-5 km)               2                   6                0
5         Corto (<1 km)               2                   6                0
…                   …             …                  …              …
157284   Medio (1-5 km)               2                   2                4
157287   Medio (1-5 km)               1                   2                4
157291    Largo (>5 km)               2                   2                4
157292    Corto (<1 km)               1                   2                4
157293    Corto (<1 km)               1                   2                4

        Start_zone_Encoded End_zone_Encoded  TrayectoCircular
1                       38               38                 1
2                       21               21                 1
3                       37               66                 0
4                       66               37                 0
5                       37               37                 1
…                     …                …                …
157284                  38               38                 1
157287                  32               75                 0
157291                  37               37                 1
157292                  49               75                 0
157293                  47               47                 1

[100561 rows x 25 columns]
```

Convierte distancia a km; redefine función de clasificación (Corto <1km, Medio 1-5km, Largo >5km); aplica y muestra df.

[ ]:
```python
# Viajes por tipo de trayecto

df['Tipo de Trayecto'].value_counts()
```

[ ]:
```
Tipo de Trayecto
Medio (1-5 km)    58946
Corto (<1 km)     28805
Largo (>5 km)     12810
Name: count, dtype: int64
```

Confirma distribución: ~59% medio, ~29% corto, ~13% largo. Uso predominantemente urbano/local.

[ ]:
```
import folium
from folium import FeatureGroup

df_map = df.dropna(subset=['Start Centroid Latitude','Start Centroid Longitude',
    'End Centroid Latitude','End Centroid Longitude','Tipo de Trayecto']).copy()

color_map = {
    'Corto (<1 km)': '#4CAF50',    # verde
    'Medio (1-5 km)': '#2196F3',   # azul
    'Largo (>5 km)': '#E91E63'     # rosa
}

center_lat = df_map['Start Centroid Latitude'].mean()
center_lon = df_map['Start Centroid Longitude'].mean()
m = folium.Map(location=[center_lat, center_lon], zoom_start=11,␣
 ↪tiles='cartodbpositron')


for tipo, sub in df_map.groupby('Tipo de Trayecto'):
    fg = FeatureGroup(name=tipo, show=True)
    col = color_map.get(tipo, '#999999')
    for _, r in sub.iterrows():
        folium.PolyLine(
            locations=[
                (r['Start Centroid Latitude'], r['Start Centroid Longitude']),
                (r['End Centroid Latitude'],   r['End Centroid Longitude'])
            ],
            color=col, weight=2.5, opacity=0.6,
            tooltip=f"{tipo} | {r['Start Community Area Name']} → {r['End␣
 ↪Community Area Name']}"
        ).add_to(fg)
    fg.add_to(m)

folium.LayerControl(collapsed=False).add_to(m)


legend_html = """
<div style="
    position: fixed; bottom: 20px; left: 20px; z-index:9999; font-size:14px;
    background: white; padding: 10px 12px; border: 1px solid #ccc;␣
 ↪border-radius: 6px;">
<b>Tipo de Trayecto</b><br>
<span style="display:inline-block;width:14px;height:4px;background:#4CAF50;
 ↪margin-right:6px"></span> Corto (&lt;1 km)<br>
```

```
<span style="display:inline-block;width:14px;height:4px;background:#2196F3;
↪margin-right:6px"></span> Medio (1-5 km)<br>
<span style="display:inline-block;width:14px;height:4px;background:#E91E63;
↪margin-right:6px"></span> Largo (&gt;5 km)
</div>
"""
m.get_root().html.add_child(folium.Element(legend_html))


m.save("mapa_trayectos_scooters.html")
m
```

```
Buffered data was truncated after reaching the output size limit.
```

Densidad alta en centro/norte de Chicago. Líneas cortas/medias dominan, concentradas en hotspots como Lake View. Útil para identificar rutas populares; circulares aparecen como puntos o líneas cortas.

[ ]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
```

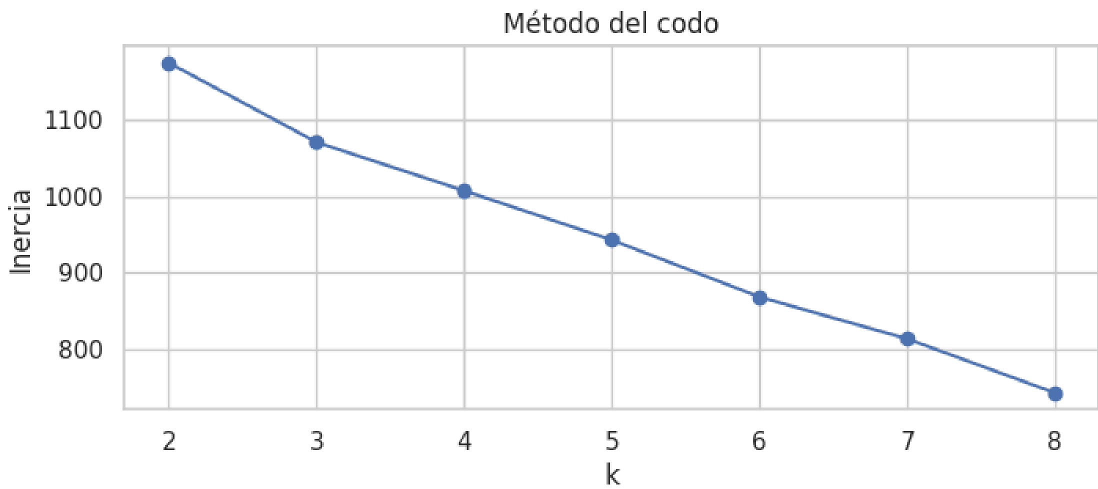
[ ]:
```
df['Date'] = pd.to_datetime(df['Date'], dayfirst=True, errors='coerce')
df['Timestamp'] = df['Date'] + pd.to_timedelta(df['Hour'], unit='h')
```

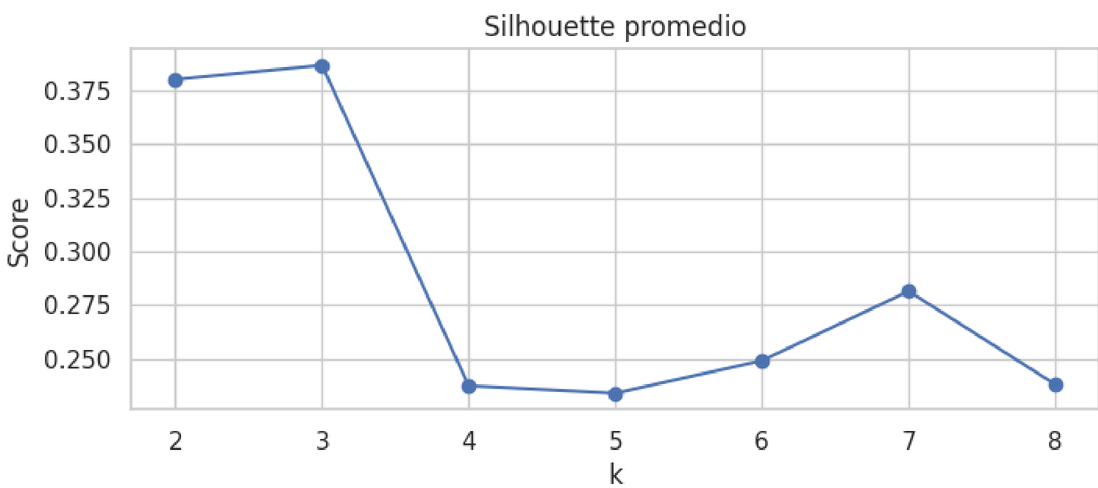
[ ]:
```
# Matriz 24-D por comunidad (conteos por hora)
M = (df.groupby(['Start Community Area Name', 'Hour'])
        .size()
        .unstack(fill_value=0)
        .reindex(columns=range(24), fill_value=0))
M.columns = [f'h{h:02d}' for h in M.columns]
M
```

[ ]:
```
# Proporciones por hora
X = M.div(M.sum(axis=1), axis=0).fillna(0)
X
```

[69]:
```
K = range(2, 9)
inertias, sils = [], []
sc_diag = StandardScaler()
Xz_diag = sc_diag.fit_transform(X)
for k_ in K:
    km_ = KMeans(n_clusters=k_, n_init=50, random_state=42)
    lab_ = km_.fit_predict(Xz_diag)
    inertias.append(km_.inertia_)
    sils.append(silhouette_score(Xz_diag, lab_))

plt.figure(figsize=(8,3))
```

```
plt.plot(K, inertias, marker='o'); plt.title('Método del codo'); plt.
  ↪xlabel('k'); plt.ylabel('Inercia'); plt.grid(True)
plt.show()
plt.figure(figsize=(8,3))
plt.plot(K, sils, marker='o'); plt.title('Silhouette promedio'); plt.
  ↪xlabel('k'); plt.ylabel('Score'); plt.grid(True)
plt.show()
```





k=4 ofrece balance (codo + silhouette razonable). Sugiere 4 patrones horarios distintos en comunidades.

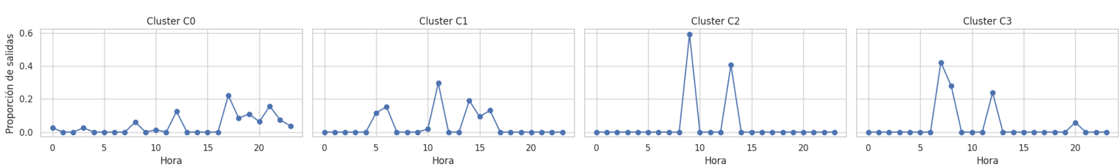
[78]:
```
k = 4  # dejamos en 4 para mayor segmentación
sc = StandardScaler()
```

```
Xz = sc.fit_transform(X)

km = KMeans(n_clusters=k, n_init=50, random_state=42).fit(Xz)
labels = pd.Series(km.labels_, index=X.index, name='Cluster')
print('Comunidades por clúster:\n', labels.value_counts().sort_index().
 ↪rename('count'))
```

```
Comunidades por clúster:
 Cluster
0    45
1    14
2     1
3     1
Name: count, dtype: int64
```

Desbalance: Cluster 0 domina (áreas mixtas), clusters 2/3 son outliers (Avalon Park, New City).

[79]:
```
# normaliza
centers_prop = pd.DataFrame(km.cluster_centers_, columns=X.columns)
centers_prop = centers_prop.clip(lower=0)
centers_prop = centers_prop.div(centers_prop.sum(axis=1), axis=0)

# media de perfiles por clúster  en X
mean_profile = (X.assign(Cluster=labels)
                  .groupby('Cluster')[X.columns].mean()
                  .rename(index=lambda i: f'C{i}'))

# gráfica de perfiles 24h (en proporciones)
to_plot = centers_prop
fig, axes = plt.subplots(1, k, figsize=(5*k, 3), sharey=True)
if k == 1: axes = [axes]
for i in range(k):
    ax = axes[i]
    ax.plot(range(24), to_plot.iloc[i].values, marker='o')
    ax.set_title(f'Cluster C{i}')
    ax.set_xlabel('Hora'); ax.grid(True)
axes[0].set_ylabel('Proporción de salidas')
plt.tight_layout(); plt.show()
```



C0: Uso diurno/tarde (recreativo). C1: Pico temprano (laboral?). C2/C3: Patrones únicos. Clustering revela tipologías: centrales/recreativas vs periféricas/laborales.

[80]:
```
comunidades_por_cluster = (labels.reset_index()
                            .groupby('Cluster')['Start Community Area Name']
                            .unique())
for c, lista in comunidades_por_cluster.items():
    print(f'\nCluster {c} ({len(lista)} comunidades):')
    print(', '.join(sorted(lista)))
```

```
Cluster 0 (45 comunidades):
ALBANY PARK, ARMOUR SQUARE, AUSTIN, AVONDALE, BELMONT CRAGIN, BRIDGEPORT,
BRIGHTON PARK, DOUGLAS, DUNNING, EAST GARFIELD PARK, EDGEWATER, ENGLEWOOD,
FOREST GLEN, GAGE PARK, GRAND BOULEVARD, HERMOSA, HUMBOLDT PARK, HYDE PARK,
IRVING PARK, JEFFERSON PARK, KENWOOD, LAKE VIEW, LINCOLN PARK, LINCOLN SQUARE,
LOGAN SQUARE, LOWER WEST SIDE, MONTCLARE, NEAR NORTH SIDE, NEAR SOUTH SIDE, NEAR
WEST SIDE, NORTH CENTER, NORTH LAWNDALE, NORTH PARK, NORWOOD PARK, OAKLAND,
PORTAGE PARK, ROGERS PARK, SOUTH LAWNDALE, SOUTH SHORE, UPTOWN, WASHINGTON PARK,
WEST GARFIELD PARK, WEST RIDGE, WEST TOWN, WOODLAWN

Cluster 1 (14 comunidades):
ASHBURN, AUBURN GRESHAM, BURNSIDE, CHATHAM, FULLER PARK, GREATER GRAND CROSSING,
LOOP, MCKINLEY PARK, PULLMAN, ROSELAND, SOUTH CHICAGO, WEST ELSDON, WEST
ENGLEWOOD, WEST LAWN

Cluster 2 (1 comunidades):
AVALON PARK

Cluster 3 (1 comunidades):
NEW CITY
```

Clusters geográficamente agrupados: C0 en norte/centro (alto uso), C1 en sur (bajo). Visualiza segregación espacial por patrones de uso.

[81]:
```
# Mapa: marcador por comunidad coloreado según cluster
#    - Centroide medio de ORIGEN por comunidad
#    - Radio del marcador proporcional al número de viajes que inician allí


centroids = (df.groupby('Start Community Area Name')
               .agg(lat=('Start Centroid Latitude', 'mean'),
                    lon=('Start Centroid Longitude', 'mean'),
                    viajes=('Start Community Area Name', 'size')))
centroids['Cluster'] = labels  # se alinea por índice (nombre de comunidad)

# Radio (tamaño)  viajes
rmin, rmax = 4, 14
v = centroids['viajes']
centroids['radius'] = rmin + (v - v.min())/(v.max()-v.min() + 1e-9) * (rmax -
 ↪rmin)
```

```python
# Colores por clúster
palette = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']
color_by_cluster = {i: palette[i % len(palette)] for i in range(k)}


m = folium.Map(
    location=[df['Start Centroid Latitude'].mean(),
              df['Start Centroid Longitude'].mean()],
    zoom_start=11, tiles='cartodbpositron'
)

for name, row in centroids.iterrows():
    folium.CircleMarker(
        location=[row['lat'], row['lon']],
        radius=float(row['radius']),
        color=color_by_cluster[int(row['Cluster'])],
        fill=True, fill_opacity=0.75, weight=1,
        popup=f"{name} • Cluster C{int(row['Cluster'])} • viajes={int(row['viajes'])}",
        tooltip=f"{name} (C{int(row['Cluster'])})"
    ).add_to(m)


legend_html = """
<div style="position: fixed; bottom: 20px; left: 20px; z-index: 9999; background: white;
            padding: 10px 12px; border: 1px solid #ccc; border-radius: 6px; font-size: 14px;">
<b>Clúster de comunidades</b><br>
""" + "".join(
    f'<div><span style="display:inline-block;width:12px;height:12px;background:{color_by_cluster[i]};'
    f'margin-right:6px;border-radius:50%"></span> C{i}</div>' for i in range(k)
) + """
<div style="margin-top:6px;">Tamaño   viajes desde la comunidad</div>
</div>
"""
m.get_root().html.add_child(folium.Element(legend_html))

m  # en notebook/colab se renderiza
# m.save("mapa_clusters_comunidades.html")  # descomenta para guardar el HTML
```

[81]:
```
<folium.folium.Map at 0x79d56fbb07d0>
```

[82]:
```python
from statsmodels.tsa.seasonal import STL
import matplotlib.pyplot as plt
```

[87]:
```python
df=pd.read_csv('Scooter_Trips_2020.csv')
df
```

[87]:
```
              Date  Hour  Trip Distance  Trip Duration Vendor  \
0       08/12/2020     5              5             21   spin
1       08/12/2020     7             13            101   spin
2       08/12/2020     7              7             50   bird
3       08/12/2020     7           3815            840   spin
4       08/12/2020     8           1444            445   spin
…              …   …            …            …    …
157289  12/12/2020    21            335            186   lime
157290  12/12/2020    21           2704           1254   lime
157291  12/12/2020    21           9257           2214   spin
157292  12/12/2020    21            878            325   lime
157293  12/12/2020    21            490            212   lime

        Start Community Area Number  End Community Area Number  \
0                              31.0                       31.0
1                               7.0                        7.0
2                              77.0                       77.0
3                               6.0                        3.0
4                               3.0                        6.0
…                              …                        …
157289                         23.0                       23.0
157290                         37.0                       61.0
157291                          6.0                        6.0
157292                         28.0                       24.0
157293                          8.0                        8.0

       Start Community Area Name End Community Area Name  \
0                 LOWER WEST SIDE         LOWER WEST SIDE
1                    LINCOLN PARK            LINCOLN PARK
2                       EDGEWATER               EDGEWATER
3                       LAKE VIEW                  UPTOWN
4                          UPTOWN               LAKE VIEW
…                            …                       …
157289              HUMBOLDT PARK           HUMBOLDT PARK
157290                FULLER PARK                NEW CITY
157291                  LAKE VIEW               LAKE VIEW
157292             NEAR WEST SIDE               WEST TOWN
157293            NEAR NORTH SIDE         NEAR NORTH SIDE

        Start Centroid Latitude  Start Centroid Longitude  \
0                     41.848335                -87.675179
1                     41.921880                -87.645647
2                     41.987114                -87.664343
3                     41.943514                -87.657498
```

```
4                       41.965435              -87.655145
…                           …                      …
157289                  41.900813              -87.723955
157290                  41.813368              -87.632599
157291                  41.943514              -87.657498
157292                  41.874254              -87.664619
157293                  41.899528              -87.633571

        End Centroid Latitude  End Centroid Longitude
0                   41.848335              -87.675179
1                   41.921880              -87.645647
2                   41.987114              -87.664343
3                   41.965435              -87.655145
4                   41.943514              -87.657498
…                       …                      …
157289              41.900813              -87.723955
157290              41.808705              -87.657612
157291              41.943514              -87.657498
157292              41.901459              -87.675568
157293              41.899528              -87.633571

[157294 rows x 13 columns]
```

[88]:
```
df['Date'] = pd.to_datetime(df['Date'])
df
```

[88]:
```
             Date  Hour  Trip Distance  Trip Duration Vendor  \
0      2020-08-12     5              5             21   spin
1      2020-08-12     7             13            101   spin
2      2020-08-12     7              7             50   bird
3      2020-08-12     7           3815            840   spin
4      2020-08-12     8           1444            445   spin
…             …    …             …              …     …
157289 2020-12-12    21            335            186   lime
157290 2020-12-12    21           2704           1254   lime
157291 2020-12-12    21           9257           2214   spin
157292 2020-12-12    21            878            325   lime
157293 2020-12-12    21            490            212   lime

        Start Community Area Number  End Community Area Number  \
0                              31.0                       31.0
1                               7.0                        7.0
2                              77.0                       77.0
3                               6.0                        3.0
4                               3.0                        6.0
…                              …                          …
157289                         23.0                       23.0
```

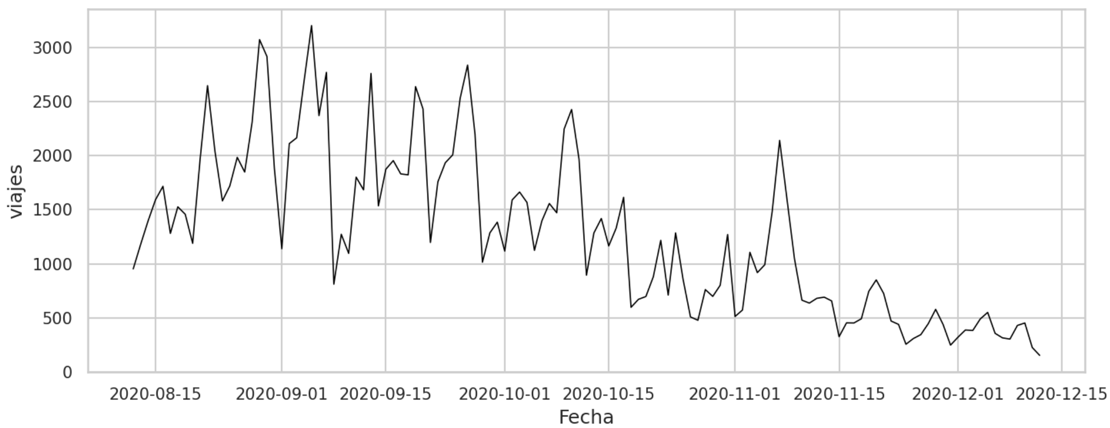
```
157290                            37.0                          61.0
157291                             6.0                           6.0
157292                            28.0                          24.0
157293                             8.0                           8.0

       Start Community Area Name End Community Area Name  \
0                 LOWER WEST SIDE         LOWER WEST SIDE
1                    LINCOLN PARK            LINCOLN PARK
2                       EDGEWATER               EDGEWATER
3                       LAKE VIEW                  UPTOWN
4                          UPTOWN               LAKE VIEW
…                               …                       …
157289              HUMBOLDT PARK           HUMBOLDT PARK
157290                 FULLER PARK                NEW CITY
157291                   LAKE VIEW               LAKE VIEW
157292             NEAR WEST SIDE               WEST TOWN
157293            NEAR NORTH SIDE         NEAR NORTH SIDE

        Start Centroid Latitude  Start Centroid Longitude  \
0                     41.848335                -87.675179
1                     41.921880                -87.645647
2                     41.987114                -87.664343
3                     41.943514                -87.657498
4                     41.965435                -87.655145
…                             …                         …
157289                41.900813                -87.723955
157290                41.813368                -87.632599
157291                41.943514                -87.657498
157292                41.874254                -87.664619
157293                41.899528                -87.633571

        End Centroid Latitude  End Centroid Longitude
0                   41.848335              -87.675179
1                   41.921880              -87.645647
2                   41.987114              -87.664343
3                   41.965435              -87.655145
4                   41.943514              -87.657498
…                           …                       …
157289              41.900813              -87.723955
157290              41.808705              -87.657612
157291              41.943514              -87.657498
157292              41.901459              -87.675568
157293              41.899528              -87.633571

[157294 rows x 13 columns]
```

[89]:
```
df.index = df['Date']
```

[90]:
```python
uso_d = df['Date'].resample('D').count()

uso_d
```

[90]:
```
Date
2020-08-12     953
2020-08-13    1180
2020-08-14    1397
2020-08-15    1593
2020-08-16    1717
              …
2020-12-08     303
2020-12-09     429
2020-12-10     452
2020-12-11     225
2020-12-12     152
Freq: D, Name: Date, Length: 123, dtype: int64
```

[91]:
```python
plt.figure(figsize=(10, 4),dpi = 150)
plt.plot(uso_d, color='black',lw = 0.8)
plt.xlabel('Fecha')
plt.ylabel('viajes')
plt.grid(True)
plt.tight_layout()
plt.tick_params(labelsize=10)
plt.show()
```



Oscilaciones semanales (fines de semana altos); declive general de agosto a diciembre (estacionalidad por frío).

[93]:
```python
from statsmodels.tsa.seasonal import STL
```

```
stl = STL(uso_d, period=7, robust=True, seasonal=21, trend=31)
result = stl.fit()
```

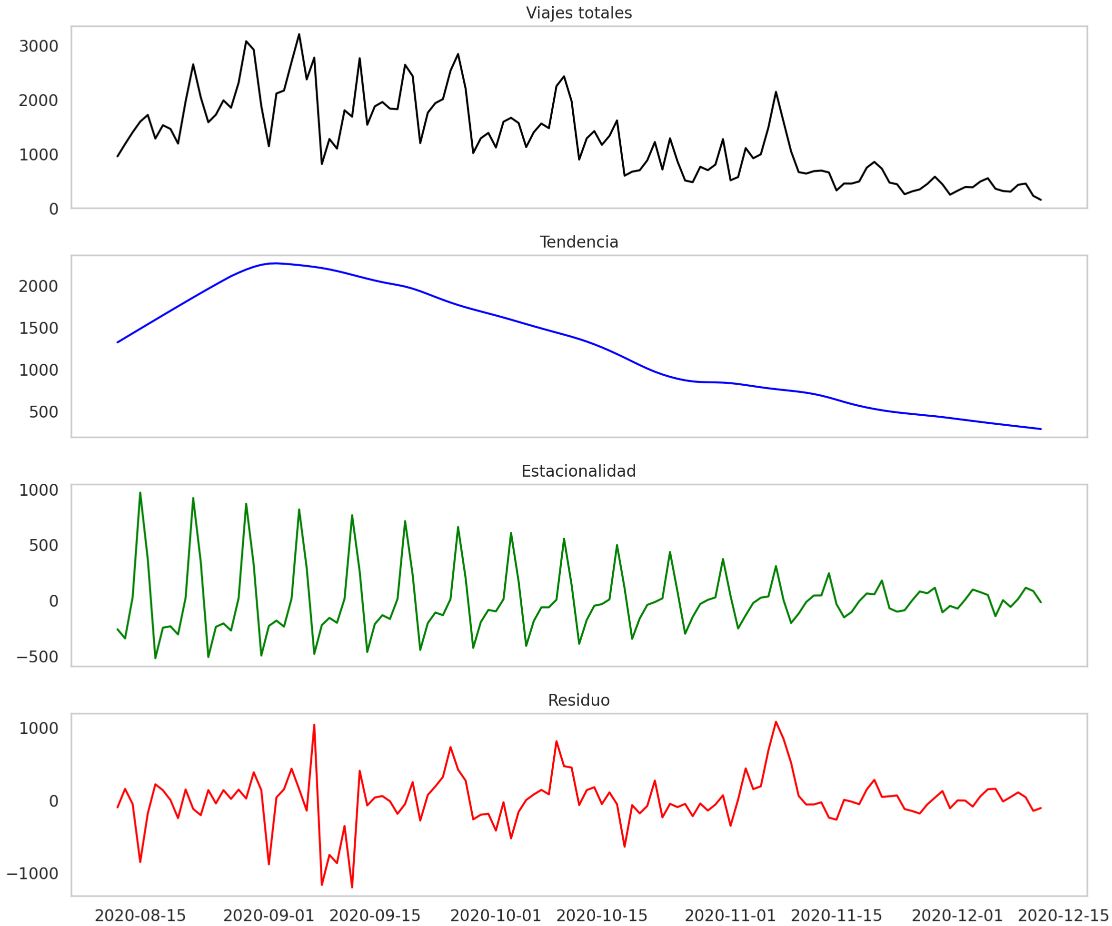
[94]:
```
df_stl = pd.DataFrame({
    'Tendencia': result.trend,
    'Estacionalidad': result.seasonal,
    'Residuo': result.resid
}, index=uso_d.index)

df_stl
```

[94]:

```
              Tendencia  Estacionalidad     Residuo
Date
2020-08-12  1319.091366     -263.007710 -103.083656
2020-08-13  1373.292254     -345.448747  152.156493
2020-08-14  1427.428628       25.992408  -56.421035
2020-08-15  1481.480890      967.469840 -855.950729
2020-08-16  1535.428422      368.496676 -186.925099
…                 …               …           …
2020-12-08   325.911935      -61.823856   38.911921
2020-12-09   315.464085       10.123380  103.412535
2020-12-10   305.058340      110.749285   36.192375
2020-12-11   294.627399       81.197977 -150.825376
2020-12-12   284.107673      -19.496821 -112.610852

[123 rows x 3 columns]
```

[95]:
```
fig, axs = plt.subplots(4, 1, figsize=(12, 10), sharex=True, dpi =200)

axs[0].plot(uso_d, label='Serie original',color = 'black')
axs[0].set_title('Viajes totales')
axs[0].tick_params(labelsize=12)
axs[0].grid()

axs[1].plot(result.trend, color='blue', label='Tendencia')
axs[1].set_title('Tendencia')
axs[1].tick_params(labelsize=12)
axs[1].grid()

axs[2].plot(result.seasonal, color='green', label='Estacionalidad')
axs[2].set_title('Estacionalidad')
axs[2].tick_params(labelsize=12)
axs[2].grid()

axs[3].plot(result.resid, color='red', label='Residuo')
axs[3].set_title('Residuo')
axs[3].tick_params(labelsize=12)
```

```
axs[3].grid()

plt.tight_layout()
plt.show()
```



Confirma estacionalidad semanal (e.g., fines de semana +). Tendencia captura declive por temporada (verano a invierno). Residuo bajo, modelo STL ajusta bien. Indica factores externos como clima afectan uso.