Prueba del modelo

Usaremos el modelo creado para hacer predicciones en un nuevo conjunto de datos.

In [3]:
```
# Cargar librerias
import pandas as pd
from datetime import timedelta
#from geopy.distance import geodesic
import numpy as np
from sklearn.preprocessing import LabelEncoder
```

Definición de los tipos de datos con base en el documento de descripción.

In [4]:
```
dtypes = {
    "trip_id": "string",
    "duration": "int8",
    "start_time": "string", # A datetime.
    "end_time":"string", #  A datetime.
    "start_lat":"float32",
    "start_lon":"float32",
    "end_lat":"float32",
    "end_lon":"float32",
    "bike_id":"string",
    "trip_route_category":"string",
    "start_station":"string",
    "end_station":"string",
}
```

In [5]:
```
# Cargar los datos
df = pd.read_csv('Data/test_set.csv',dtype=dtypes)
```

In [6]:
```
df
```

Out[6]:

	trip_id	duration	start_time	end_time	start_lat	start_lon	end_lat	
0	17059130	12	1/1/2017 0:24	1/1/2017 0:36	34.058319	-118.246094	34.058319	-
1	17059129	17	1/1/2017 0:28	1/1/2017 0:45	34.049980	-118.247162	34.043732	-
2	17059126	20	1/1/2017 0:39	1/1/2017 0:59	34.063389	-118.236160	34.044159	-
3	17059125	12	1/1/2017 0:43	1/1/2017 0:55	34.048851	-118.246422	34.050140	-
4	17061379	48	1/1/2017 0:56	1/1/2017 1:44	34.049198	-118.252831	34.049198	-
...	...	...	...	...	...	...	...	
569881	179408277	19	12/31/2021 23:29	12/31/2021 23:48	34.040989	-118.255798	34.041130	-
569882	179408276	8	12/31/2021 23:40	12/31/2021 23:48	34.044701	-118.252441	34.051941	-
569883	179409877	43	12/31/2021 23:47	1/1/2022 0:30	34.044701	-118.252441	34.044701	-
569884	179409876	42	12/31/2021 23:48	1/1/2022 0:30	34.044701	-118.252441	34.044701	-
569885	179492474	-96	12/31/2021 23:48	1/2/2022 12:56	34.051941	-118.243530	34.047440	-

569886 rows × 12 columns

Transformar datos para poder alimentarlos al modelo

Similares a los que hicimos en el proceso de EDA y de construcción del modelo

In [7]:
```
df.isna().sum()
```

Out[7]:
```
trip_id                  0
duration                 0
start_time               0
end_time                 0
start_lat             4622
start_lon             4622
end_lat              14891
end_lon              14891
bike_id                  0
trip_route_category      0
start_station            0
end_station              0
dtype: int64
```

In [8]:
```
df['start_time'] = pd.to_datetime(df['start_time'], format='mixed')
df['end_time'] = pd.to_datetime(df['end_time'], format='mixed')
```

Variable duration

In [9]:
```
df[df['duration']<0]
```

Out[9]:

	trip_id	duration	start_time	end_time	start_lat	start_lon	end_lat	
60	17096115	-96	2017-01-01 13:42:00	2017-01-01 16:22:00	34.050911	-118.240967	34.050911	-11
66	17098623	-60	2017-01-01 14:06:00	2017-01-01 17:22:00	34.035679	-118.270813	34.045422	-11
157	17151394	-102	2017-01-02 12:40:00	2017-01-02 15:14:00	34.048401	-118.260948	34.048401	-11
158	17153956	-51	2017-01-02 13:13:00	2017-01-02 16:38:00	34.039871	-118.250038	34.039871	-11
159	17153954	-53	2017-01-02 13:15:00	2017-01-02 16:38:00	34.039871	-118.250038	34.039871	-11
...	...	...	...	...	...	...	...	
569802	179403792	-79	2021-12-31 15:39:00	2021-12-31 18:36:00	34.014309	-118.491341	34.056610	-11
569836	179420708	-8	2021-12-31 17:09:00	2022-01-01 05:49:00	33.998341	-118.461014	33.998341	-11
569848	179482974	-96	2021-12-31 18:11:00	2022-01-02 08:13:00	33.995560	-118.481552	NaN	
569863	179482938	-96	2021-12-31 19:49:00	2022-01-02 08:11:00	34.098000	-118.300468	NaN	
569885	179492474	-96	2021-12-31 23:48:00	2022-01-02 12:56:00	34.051941	-118.243530	34.047440	-11

17446 rows × 12 columns

In [10]:
```
df['new_duration'] = df['end_time'] - df['start_time']
```

In [11]:
```
df.loc[df['new_duration'] <= timedelta(minutes=0), 'new_duration'] = df['duration']
```

Imputar valores de duration

In [12]:
```
df['duration']= df['new_duration'].dt.seconds / 60
```

In [13]:
```
moda = df['duration'].mode()[0]
df.loc[df['duration'] <= 0, 'duration'] = moda
```

In [14]:
```
df.isna().sum()
```

Out[14]:
```
trip_id                   0
duration                  0
start_time                0
end_time                  0
start_lat              4622
start_lon              4622
end_lat               14891
end_lon               14891
bike_id                   0
trip_route_category       0
start_station             0
end_station               0
new_duration              0
dtype: int64
```

Generación de variables

In [17]:
```
df['start_hour'] = df['start_time'].dt.hour
df['start_day'] = df['start_time'].dt.day
df['start_month'] = df['start_time'].dt.month
df['start_year'] = df['start_time'].dt.year
df['start_weekday'] = df['start_time'].dt.weekday
```

In [18]:
```
# Transformar variables categóricas
cat_columns = ['start_station', 'end_station', 'trip_route_category']
for col in cat_columns:
    df[col] = LabelEncoder().fit_transform(df[col])
```

In [19]:
```
features = ['start_station','end_station','start_hour','start_year', 'start_month',
X = df[features]
```

Datos para usar con el modelo

In [21]:
```
X
```

Out[21]:

	start_station	end_station	start_hour	start_year	start_month	start_day	start_wee
0	20	20	0	2017	1	1	
1	19	11	0	2017	1	1	
2	53	42	0	2017	1	1	
3	21	62	0	2017	1	1	
4	50	50	0	2017	1	1	
...	...	...	...	...	...	...	
569881	25	7	23	2021	12	31	
569882	23	22	23	2021	12	31	
569883	23	23	23	2021	12	31	
569884	23	23	23	2021	12	31	
569885	22	287	23	2021	12	31	

569886 rows × 9 columns

Cargar el modelo

In [27]:
```
import joblib

model = joblib.load('Modelo/lightgbm_model.pkl')

# Verificar
print(model)
```

```
LGBMClassifier(n_jobs=-1, random_state=42)
```

Correr el modelo para generar las predicciones de la variable objetivo

In [28]:
```
y_pred = model.predict(X)
```

In [ ]:
```
# Retransformar nombre de clases de variable objetivo
label_mapping = {'Annual Pass': 0, 'Flex Pass': 1, 'Monthly Pass': 2, 'One Day Pass
inverse_mapping = {v: k for k, v in label_mapping.items()}
y_pred_labels = [inverse_mapping[pred] for pred in y_pred]
```

In [30]:
```
# Asignar variable objetivo
df['passholder_type']=y_pred_labels
```

In [32]:
```
# Ver resultados
df[['trip_id','passholder_type']]
```

Out[32]:

	trip_id	passholder_type
0	17059130	Walk-up
1	17059129	Walk-up
2	17059126	Walk-up
3	17059125	Walk-up
4	17061379	Walk-up
...	...	...
569881	179408277	Monthly Pass
569882	179408276	Monthly Pass
569883	179409877	Walk-up
569884	179409876	Walk-up
569885	179492474	Walk-up

569886 rows × 2 columns

In [34]:
```
# Guardar datos como csv para evaluación en Kaggel
df[['trip_id','passholder_type']].to_csv('Data/set_final.csv', index=False)
```