

Universidad Austral
Maestría en Explotación de Datos y Gestión del Conocimiento
Cohorte 2023 - 2024
Modalidad Virtual



Análisis de Series Temporales
Trabajo Práctico N.º 2
CÓDIGO UTILIZADO

Alumnos

Cablinski Pablo
Carreño Giscagré Francisco Salvador
Griffiths Waldo
Parra Iván Javier



Ciclo Académico 2024

PROPHET CV GRIDSEARCH

TRAIN 90%

TEST 10%

```
In [1]: # Python
```

```
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
import matplotlib.pyplot as plt
from prophet.plot import plot_cross_validation_metric
```

```
In [2]:
```

```
df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-"
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPIA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
titulos = ["CHU_COPIA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

```
In [3]:
```

```
# TRAIN TEST
train_copa = dataframes[0].iloc[:round(len(dataframes[0])*0.8)]
test_copa = dataframes[0].iloc[round(len(dataframes[0])*0.8):]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[:round(len(dataframes[1])*0.8)]
test_recursos = dataframes[1].iloc[round(len(dataframes[1])*0.8):]
print(f"Recursos: train({train_recursos.shape}), test({test_recursos.shape})")

train_regalias = dataframes[2].iloc[:round(len(dataframes[2])*0.8)]
test_regalias = dataframes[2].iloc[round(len(dataframes[2])*0.8):]
print(f"Regalias: train({train_regalias.shape}), test({test_regalias.shape})")

dataframes_train = [ train_copa, train_recursos, train_regalias ]
dataframes_test = [ test_copa, test_recursos, test_regalias ]
```

```
Coparticipacion: train((1275,)), test((319,))
```

```
Recursos: train((1626,)), test((406,))
```

```
Regalias: train((460,)), test((115,))
```

```
In [ ]: parametros = []
```

```
for i, df in enumerate(dataframes_train):
    df_train = df.to_frame().reset_index(drop=False)
    df_train.columns = ["ds", "y"]

    param_grid = {
        'changepoint_prior_scale': [0.001, 0.01, 0.1, 0.5],
        'seasonality_prior_scale': [0.01, 0.1, 1.0, 10.0],
        'daily_seasonality': [True, False],
        'yearly_seasonality': [True, False],
        'holidays_prior_scale': [0.01, 0.1, 1, 10],
        'seasonality_mode': ['additive', 'multiplicative'],
        'changepoint_range': [0.8, 0.9, 0.95]
    }

    # Generate all combinations of parameters
    all_params = [dict(zip(param_grid.keys(), v)) for v in itertools.product(*pa
```

```

rmses = [] # Store the RMSEs for each params here

# Use cross validation to evaluate all parameters
for params in all_params:
    m = Prophet(**params).fit(df_train) # Fit model with given params
    df_cv = cross_validation(m, period='180 days', horizon = '365 days')
    df_p = performance_metrics(df_cv, rolling_window=1)
    rmses.append(df_p['rmse'].values[0])

# Find the best parameters
tuning_results = pd.DataFrame(all_params)
tuning_results[ 'rmse' ] = rmses
best_params = all_params[np.argmin(rmses)]
parametros.append({ "df": dataframes_train[i].name, "best_params": best_params })
print(f"Entrenamiento de {dataframes_train[i].name} finalizado")

```

In [12]: `pd.DataFrame(parametros)`

	df	best_params
0	CHU_COPA_AJUST	{'changepoint_prior_scale': 0.1, 'seasonality_prior_scale': 0.1, 'changepoint_smoothing_prior_scale': 0.05, 'holidays_prior_scale': 0.05}
1	CHU_REC_PROPIOS_AJUST	{'changepoint_prior_scale': 0.001, 'seasonality_prior_scale': 0.001, 'changepoint_smoothing_prior_scale': 0.001, 'holidays_prior_scale': 0.001}
2	CHU_REGALIAS_AJUST	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 0.5, 'changepoint_smoothing_prior_scale': 0.05, 'holidays_prior_scale': 0.05}

In [13]: `pd.DataFrame(pd.DataFrame(parametros)).to_csv("best_params.csv", index=False)`

In []:

In [1]:

```
# Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
import matplotlib.pyplot as plt
from prophet.plot import plot_cross_validation_metric
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
import funciones
```

In [2]:

```
df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-"
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0],
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

In [3]:

```
# TRAIN TEST
n_train = 0.9
train_copa = dataframes[0].iloc[:round(len(dataframes[0])*n_train)]
test_copa = dataframes[0].iloc[round(len(dataframes[0])*n_train):]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[:round(len(dataframes[1])*n_train)]
test_recursos = dataframes[1].iloc[round(len(dataframes[1])*n_train):]
print(f"Recursos: train({train_recursos.shape}), test({test_recursos.shape})")

train_regalias = dataframes[2].iloc[:round(len(dataframes[2])*n_train)]
test_regalias = dataframes[2].iloc[round(len(dataframes[2])*n_train):]
print(f"Regalias: train({train_regalias.shape}), test({test_regalias.shape})")

dataframes_train = [ train_copa, train_recursos, train_regalias ]
dataframes_test = [ test_copa, test_recursos, test_regalias ]
```

Coparticipacion: train((1584,)), test((176,))

Recursos: train((1995,)), test((222,))

Regalias: train((1985,)), test((221,))

In []:

```
from prophet.diagnostics import cross_validation, performance_metrics

fourier_y_list = []
for df in dataframes_train:
    name = df.name
    df = df.to_frame()
    df = df.reset_index()
    df.columns = ['ds', 'y']
    for fourier in [3, 5, 7, 10]:
        model = Prophet(yearly_seasonality=False)
```

```
model.add_seasonality(name='yearly', period=365.25, fourier_order=fourier)
model.fit(df)

df_cv = cross_validation(model, initial='730 days', period='180 days', h=180)
df_p = performance_metrics(df_cv)
print(f'Fourier Order: {fourier}, mae: {df_p["mae"].mean()}')
fourier_y_list.append({
    'name': name,
    'fourier_order': fourier,
    'mae': df_p["mae"].mean(),
    'mse': df_p["mse"].mean(),
    'rmse': df_p["rmse"].mean(),
})

})
```

```
In [ ]: from prophet.diagnostics import cross_validation, performance_metrics
fourier_m_list = []
for df in dataframes_train:
    name = df.name
    df = df.to_frame()
    df = df.reset_index()
    df.columns = ['ds', 'y']
    for fourier in [3, 5, 7, 10]:
        model = Prophet(yearly_seasonality=False)
        model.add_seasonality(name='monthly', period=30.5, fourier_order=fourier)
        model.fit(df)

        df_cv = cross_validation(model, initial='730 days', period='180 days', h=180)
        df_p = performance_metrics(df_cv)
        print(f'Fourier Order: {fourier}, mae: {df_p["mae"].mean()}')
        fourier_m_list.append({
            'name': name,
            'fourier_order': fourier,
            'mae': df_p["mae"].mean(),
            'mse': df_p["mse"].mean(),
            'rmse': df_p["rmse"].mean(),
        })

    })
```

```
In [24]: pd.DataFrame(fourier_m_list)
```

Out[24]:

	name	fourier_order	mae	mse	rm
0	CHU_COPA_AJUST	3	1,295,550.58	3,368,417,717,852.50	1,829,278.
1	CHU_COPA_AJUST	5	1,292,136.70	3,382,102,160,158.99	1,833,572.
2	CHU_COPA_AJUST	7	1,289,267.78	3,403,619,136,925.37	1,839,180.
3	CHU_COPA_AJUST	10	1,292,343.45	3,435,217,624,484.17	1,847,424.
4	CHU_REC_PROPIOS_AJUST	3	875,832.51	1,354,696,658,529.47	1,160,714.
5	CHU_REC_PROPIOS_AJUST	5	876,294.82	1,360,751,514,817.23	1,163,577.
6	CHU_REC_PROPIOS_AJUST	7	884,354.91	1,384,814,129,416.05	1,173,991.
7	CHU_REC_PROPIOS_AJUST	10	886,719.39	1,389,622,924,478.05	1,175,983.
8	CHU_REGALIAS_AJUST	3	2,981,291.15	33,617,382,219,021.43	5,786,977.
9	CHU_REGALIAS_AJUST	5	2,881,989.42	34,018,833,939,631.12	5,820,355.
10	CHU_REGALIAS_AJUST	7	2,887,115.68	34,332,503,235,722.89	5,847,218.
11	CHU_REGALIAS_AJUST	10	2,896,604.65	34,623,962,264,283.60	5,872,018.

◀ ▶

In [28]:

```
pd.options.display.float_format = '{:.2f}'.format
pd.DataFrame(fourier_y_list)
```

Out[28]:

	name	fourier_order	mae	mse	rm
0	CHU_COPA_AJUST	3	1,659,037.26	4,493,518,123,189.03	2,114,628.
1	CHU_COPA_AJUST	5	1,659,235.07	4,492,317,272,786.19	2,114,568.
2	CHU_COPA_AJUST	7	1,662,586.82	4,503,776,502,703.05	2,117,144.
3	CHU_COPA_AJUST	10	1,662,410.64	4,524,420,159,270.77	2,120,638.
4	CHU_REC_PROPIOS_AJUST	3	1,007,027.71	1,554,772,294,789.62	1,244,465.
5	CHU_REC_PROPIOS_AJUST	5	1,006,999.60	1,550,073,308,813.71	1,242,539.
6	CHU_REC_PROPIOS_AJUST	7	1,005,906.47	1,551,414,785,218.95	1,242,870.
7	CHU_REC_PROPIOS_AJUST	10	1,004,277.36	1,548,804,571,628.49	1,241,865.
8	CHU_REGALIAS_AJUST	3	2,975,730.22	36,804,980,074,407.97	6,054,561.
9	CHU_REGALIAS_AJUST	5	2,981,596.93	36,810,240,630,045.30	6,054,990.
10	CHU_REGALIAS_AJUST	7	2,961,931.77	36,818,174,730,371.95	6,055,578.
11	CHU_REGALIAS_AJUST	10	2,970,283.20	36,865,017,859,487.95	6,059,508.

◀ ▶

In [35]:

```
def get_best_fourier_orders(fourier_list):
    best_orders = {}
    for item in fourier_list:
        name = item['name']
        if name not in best_orders or item['rmse'] < best_orders[name]['rmse']:
            best_orders[name] = {
```

```

        'fourier_order': item['fourier_order'],
        'rmse': item['rmse']
    }
    return best_orders

best_fourier_orders_y = get_best_fourier_orders(fourier_y_list)
best_fourier_orders_m = get_best_fourier_orders(fourier_m_list)

print("Best Fourier Orders (Yearly):", best_fourier_orders_y)
print("Best Fourier Orders (Monthly):", best_fourier_orders_m)
result = pd.DataFrame([best_fourier_orders_y, best_fourier_orders_m]).T
result.columns = ['Fourier_yearly', 'Fourier_monthly']
result

```

Best Fourier Orders (Yearly): {'CHU_COPA_AJUST': {'fourier_order': 5, 'rmse': 214568.5994672766}, 'CHU_REC_PROPIOS_AJUST': {'fourier_order': 10, 'rmse': 1241865.2497155124}, 'CHU_REGALIAS_AJUST': {'fourier_order': 3, 'rmse': 6054561.279307909}}

Best Fourier Orders (Monthly): {'CHU_COPA_AJUST': {'fourier_order': 3, 'rmse': 1829278.0979960766}, 'CHU_REC_PROPIOS_AJUST': {'fourier_order': 3, 'rmse': 1160714.3127307887}, 'CHU_REGALIAS_AJUST': {'fourier_order': 3, 'rmse': 5786977.88651003}}

Out[35]:

	Fourier_yearly	Fourier_monthly
CHU_COPA_AJUST	{'fourier_order': 5, 'rmse': 214568.5994672766}	{'fourier_order': 3, 'rmse': 1829278.0979960766}
CHU_REC_PROPIOS_AJUST	{'fourier_order': 10, 'rmse': 1241865.2497155124}	{'fourier_order': 3, 'rmse': 1160714.3127307887}
CHU_REGALIAS_AJUST	{'fourier_order': 3, 'rmse': 6054561.279307909}	{'fourier_order': 3, 'rmse': 5786977.886510033}

In [36]: `result.to_csv("best_fourier_orders.csv")`

In []:

PROPHET

TRAIN 90%

TEST 10%

In [7]:

```
# Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
import matplotlib.pyplot as plt
from prophet.plot import plot_cross_validation_metric
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
import funciones
```

In [8]:

```
df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-"
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0],
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

In [9]:

```
# TRAIN TEST
n_train = 0.9
train_copa = dataframes[0].iloc[:round(len(dataframes[0])*n_train)]
test_copa = dataframes[0].iloc[round(len(dataframes[0])*n_train):]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[:round(len(dataframes[1])*n_train)]
test_recursos = dataframes[1].iloc[round(len(dataframes[1])*n_train):]
print(f"Recursos: train({train_recursos.shape}), test({test_recursos.shape})")

train_regalias = dataframes[2].iloc[:round(len(dataframes[2])*n_train)]
test_regalias = dataframes[2].iloc[round(len(dataframes[2])*n_train):]
print(f"Regalias: train({train_regalias.shape}), test({test_regalias.shape})")

dataframes_train = [ train_copa, train_recursos, train_regalias ]
dataframes_test = [ test_copa, test_recursos, test_regalias ]
```

Coparticipacion: train((1584,)), test((176,))

Recursos: train((1995,)), test((222,))

Regalias: train((1985,)), test((221,))

In [19]: model.predict(future)

Out[19]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper
0	2018-01-02	2.244932e+06	-669003.795682	4.370992e+06	2.244932e+06	2.244932e+06
1	2018-01-03	2.244625e+06	-295740.561883	4.843847e+06	2.244625e+06	2.244625e+06
2	2018-01-04	2.244318e+06	-227764.824909	4.710549e+06	2.244318e+06	2.244318e+06
3	2018-01-05	2.244011e+06	-241790.890588	4.897786e+06	2.244011e+06	2.244011e+06
4	2018-01-08	2.243090e+06	-709402.692045	4.526881e+06	2.243090e+06	2.243090e+06
...
1755	2024-09-24	2.881273e+06	46619.465574	5.376678e+06	2.870376e+06	2.892725e+06
1756	2024-09-25	2.881827e+06	504211.934660	5.675143e+06	2.870773e+06	2.893353e+06
1757	2024-09-26	2.882382e+06	324626.675420	5.514598e+06	2.871118e+06	2.893978e+06
1758	2024-09-27	2.882937e+06	334686.836801	5.772139e+06	2.871469e+06	2.894602e+06
1759	2024-09-30	2.884602e+06	137319.920679	5.109929e+06	2.873045e+06	2.896327e+06

1760 rows × 22 columns

In []:

```

results_train_test = []
predictions_test = []
from prophet.make_holidays import make_holidays_df
best_params = pd.read_csv("prophet_best_params.csv")
best_fourier = pd.read_csv("best_fourier_orders.csv")

for i, df_train in enumerate(dataframes_train):
    df_train = df_train.to_frame()
    df_train.reset_index(inplace=True)
    df_train.columns = ["ds", "y"]
    df_test = dataframes_test[i]
    params = eval(best_params.iloc[i]["best_params"])
    year_list = dataframes[i].index.year.unique()
    holidays = make_holidays_df(year_list=year_list, country='AR')
    model = Prophet(**params,holidays=holidays)

    fourier_yearly = eval(best_fourier.iloc[i]["Fourier_yearly"])["fourier_order"]
    fourier_monthly = eval(best_fourier.iloc[i]["Fourier_monthly"])["fourier_ord
model.add_seasonality(name='monthly', period=30.5, fourier_order=fourier_mon
model.add_seasonality(name='yearly', period=365.25, fourier_order=fourier_ye

model.fit(df_train)
fechas = pd.date_range(start=df_test.index.min(), end=df_test.index.max(), f
future = model.make_future_dataframe(periods=len(fechas), freq='B')

```

```

pred_test = model.predict(future)
pred_test.index = pred_test["ds"]
pred_test = pred_test["yhat"]
pred_test = pred_test[-len(df_test):]

predictions_test.append(pred_test)
# Cálculo del MSE en el conjunto de prueba
mape_test = mean_absolute_percentage_error(df_test, pred_test)
mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(df_test))
mse_test = mean_squared_error(df_test, pred_test)
mae_test = mean_absolute_error(df_test, pred_test)
rmse = np.sqrt(mean_squared_error(df_test, pred_test))
results_train_test.append({
    "model": model,
    "name": titulos[i],
    "len_train": len(df_train),
    "len_test": len(df_test),
    "mape_test": mape_test,
    "mse_test": mse_test,
    "mape_mean": mape_mean,
    "mae_test": mae_test,
    "rmse": rmse
})

```

```

15:47:26 - cmdstanpy - INFO - Chain [1] start processing
15:47:26 - cmdstanpy - INFO - Chain [1] done processing
15:47:27 - cmdstanpy - INFO - Chain [1] start processing
15:47:27 - cmdstanpy - INFO - Chain [1] done processing
15:47:28 - cmdstanpy - INFO - Chain [1] start processing
15:47:28 - cmdstanpy - INFO - Chain [1] done processing

```

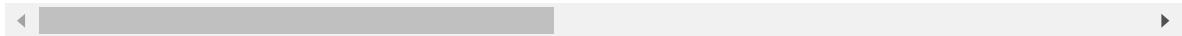
```

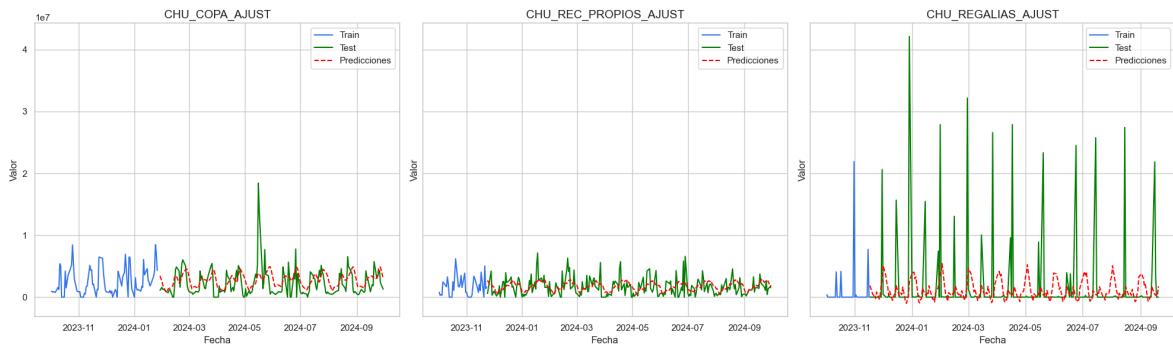
In [54]: pd.options.display.float_format = '{:.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=dataframes_train,
    dataframes_test=dataframes_test,
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2023-10-01'
))

```

	model	name	len_train	len_test	
0	<prophet.forecaster.Prophet object at 0x000002...	CHU_COPA_AJUST	1584	176	958,736,396,89
1	<prophet.forecaster.Prophet object at 0x000002...	CHU_REC_PROPIOS_AJUST	1995	222	552,112,145,06
2	<prophet.forecaster.Prophet object at 0x000002...	CHU_REGALIAS_AJUST	1985	221	3,911,866,127,92





None

In []:

XGBOOST

TRAIN 90%

TEST 10%

In [4]:

```
# Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
import matplotlib.pyplot as plt
from prophet.plot import plot_cross_validation_metric
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
import funciones
import xgboost as xgb
```

In [6]:

```
df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-"
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0],
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

In [7]:

```
def extract_time_features(index):
    return pd.Series({
        'dayofweek': index.dayofweek,
        'quarter': index.quarter,
        'month': index.month,
        'year': index.year,
        'dayofyear': index.dayofyear,
        'dayofmonth': index.day,
        'weekofyear': index.isocalendar().week
    })
def add_lags(df, titulo):
    target_map = df[titulo].to_dict()
    df['lag1'] = (df.index - pd.Timedelta('364 days')).map(target_map) # df_1['P'
    df['lag2'] = (df.index - pd.Timedelta('728 days')).map(target_map) # df_1['P'
    df['lag3'] = (df.index - pd.Timedelta('1092 days')).map(target_map) # df_1['P'
    return df

for i in range(len(dataframes)):
    time_features = dataframes[i].index.to_series().apply(extract_time_features)
    dataframes[i] = pd.concat([dataframes[i], time_features], axis=1)
    dataframes[i] = add_lags(dataframes[i], titulos[i])
```

In [8]:

```
# TRAIN TEST
n_train = 0.9
train_copa = dataframes[0].iloc[:round(len(dataframes[0])*n_train)]
test_copa = dataframes[0].iloc[round(len(dataframes[0])*n_train):]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")
```

```

train_recursos = dataframes[1].iloc[:round(len(dataframes[1])*n_train)]
test_recursos = dataframes[1].iloc[round(len(dataframes[1])*n_train):]
print(f'Recursos: train({train_recursos.shape}), test({test_recursos.shape})')

train_regalias = dataframes[2].iloc[:round(len(dataframes[2])*n_train)]
test_regalias = dataframes[2].iloc[round(len(dataframes[2])*n_train):]
print(f'Regalias: train({train_regalias.shape}), test({test_regalias.shape})')

dataframes_train = [ train_copa, train_recursos, train_regalias ]
dataframes_test = [ test_copa, test_recursos, test_regalias ]

```

Coparticipacion: train((1584, 11)), test((176, 11))
 Recursos: train((1995, 11)), test((222, 11))
 Regalias: train((1985, 11)), test((221, 11))

```

In [ ]: results_train_test = []
predictions_test = []
best_params = pd.read_csv("xgb_best_params.csv", sep=";")

for i, df_train in enumerate(dataframes_train):

    params = eval(best_params.iloc[i]["best_params"])

    FEATURES = ['dayofweek', 'quarter', 'month', 'year', 'dayofyear' , 'dayofmon'
    TARGET = titulos[i]

    df_test = dataframes_test[i][TARGET]
    model = xgb.XGBRegressor(**params )

    X_train = dataframes_train[i][FEATURES]
    y_train = dataframes_train[i][TARGET]
    X_test= dataframes_test[i][FEATURES]
    y_test= dataframes_test[i][TARGET]

    model.fit(
        X_train, y_train,
        eval_set=[(X_test, y_test)])
    )

    pred_test = model.predict(X_test)
    pred_test = pd.Series(pred_test, index=dataframes_test[i].index)
    predictions_test.append(pred_test)

    # Cálculo del MSE en el conjunto de prueba
    mape_test = mean_absolute_percentage_error(df_test, pred_test)
    mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(df_test))
    mse_test = mean_squared_error(df_test, pred_test)
    mae_test = mean_absolute_error(df_test, pred_test)
    rmse = np.sqrt(mean_squared_error(df_test, pred_test))
    results_train_test.append({
        "model": model,
        "name": titulos[i],
        "len_train": len(df_train),
        "len_test": len(df_test),
        "mape_test": mape_test,
        "mse_test":mse_test,
        "mape_mean": mape_mean,
        "mae_test": mae_test,
        "rmse": rmse
    })

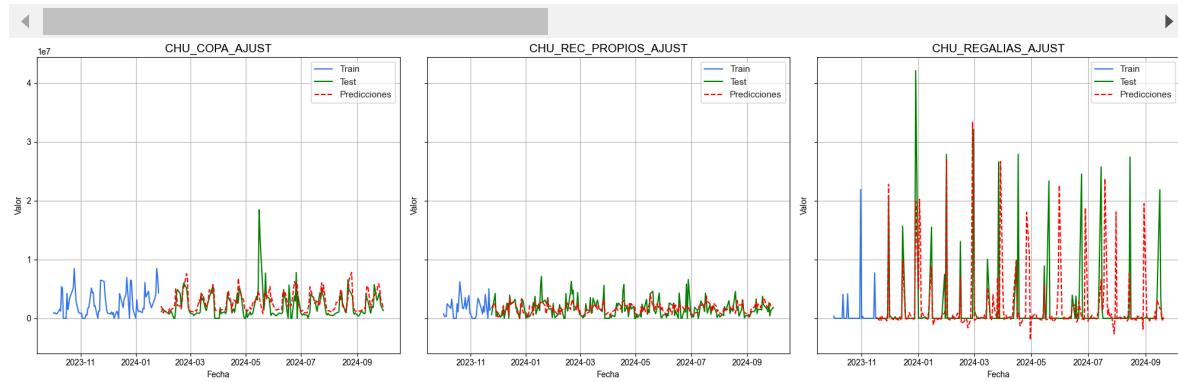
```

```
In [10]: pd.options.display.float_format = '{:.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=[pd.Series(dataframes_train[i][titulos[i]], index=dataframe
    dataframes_test=[pd.Series(dataframes_test[i][titulos[i]], index=dataframes_
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2023-10-01'
)))

```

	model	name	len_train	len_test
0	XGBRegressor(base_score=None, booster=None, ca...	CHU_COPA_AJUST	1584	176 696,802,90
1	XGBRegressor(base_score=None, booster=None, ca...	CHU_REC_PROPIOS_AJUST	1995	222 390,317,60
2	XGBRegressor(base_score=None, booster=None, ca...	CHU_REGALIAS_AJUST	1985	221 6,052,706,5



None

```
In [1]: # Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
import matplotlib.pyplot as plt
from prophet.plot import plot_cross_validation_metric
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
import funciones
```

```
In [2]: df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-"
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0],
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

```
In [3]: def extract_time_features(index):
    return pd.Series({
        'dayofweek': index.dayofweek,
        'quarter': index.quarter,
        'month': index.month,
        'year': index.year,
        'dayofyear': index.dayofyear,
        'dayofmonth': index.day,
        'weekofyear': index.isocalendar().week
    })
def add_lags(df, titulo):
    target_map = df[titulo].to_dict()
    df['lag1'] = (df.index - pd.Timedelta('364 days')).map(target_map) # df_1['P']
    df['lag2'] = (df.index - pd.Timedelta('728 days')).map(target_map) # df_1['P']
    df['lag3'] = (df.index - pd.Timedelta('1092 days')).map(target_map) # df_1['P'
    return df

for i in range(len(dataframes)):
    time_features = dataframes[i].index.to_series().apply(extract_time_features)
    dataframes[i] = pd.concat([dataframes[i], time_features], axis=1)
    dataframes[i] = add_lags(dataframes[i], titulos[i])
```

```
In [4]: # TRAIN TEST
n_train = 0.9
train_copa = dataframes[0].iloc[:round(len(dataframes[0])*n_train)]
test_copa = dataframes[0].iloc[round(len(dataframes[0])*n_train):]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[:round(len(dataframes[1])*n_train)]
```

```

test_recursos = dataframes[1].iloc[round(len(dataframes[1])*n_train):]
print(f'Recursos: train({train_recursos.shape}), test({test_recursos.shape})')

train_Regalias = dataframes[2].iloc[:round(len(dataframes[2])*n_train)]
test_Regalias = dataframes[2].iloc[round(len(dataframes[2])*n_train):]
print(f'Regalias: train({train_Regalias.shape}), test({test_Regalias.shape})')

dataframes_train = [ train_copa, train_recursos, train_Regalias ]
dataframes_test = [ test_copa, test_recursos, test_Regalias ]

```

Coparticipacion: train((1584, 11)), test((176, 11))

Recursos: train((1995, 11)), test((222, 11))

Regalias: train((1985, 11)), test((221, 11))

```

In [ ]: results_train_test = []
predictions_test = []
best_params = pd.read_csv("lgbm_best_params.csv",sep=";")
import lightgbm as lgb

for i, df_train in enumerate(dataframes_train):

    params = eval(best_params.iloc[i]["best_params"])

    FEATURES = ['dayofweek', 'quarter', 'month', 'year', 'dayofyear' , 'dayofmon'
    TARGET = titulos[i]

    df_test = dataframes_test[i][TARGET]
    model = lgb.LGBMRegressor(**params )

    X_train = dataframes_train[i][FEATURES]
    y_train = dataframes_train[i][TARGET]
    X_test= dataframes_test[i][FEATURES]
    y_test= dataframes_test[i][TARGET]

    model.fit(df_train[FEATURES], df_train[TARGET],
              eval_set=[(X_train, y_train), (X_test, y_test)], eval_metric="rmse"

    pred_test = model.predict(dataframes_test[i][FEATURES])
    pred_test = pd.Series(pred_test, index=dataframes_test[i].index)
    predictions_test.append(pred_test)

    # Cálculo del MSE en el conjunto de prueba
    mape_test = mean_absolute_percentage_error(df_test, pred_test)
    mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(df_test))
    mse_test = mean_squared_error(df_test, pred_test)
    mae_test = mean_absolute_error(df_test, pred_test)
    rmse = np.sqrt(mean_squared_error(df_test, pred_test))
    results_train_test.append({
        "model": model,
        "name": titulos[i],
        "len_train": len(df_train),
        "len_test": len(df_test),
        "mape_test": mape_test,
        "mse_test":mse_test,
        "mape_mean": mape_mean,
        "mae_test": mae_test,
        "rmse": rmse
    })

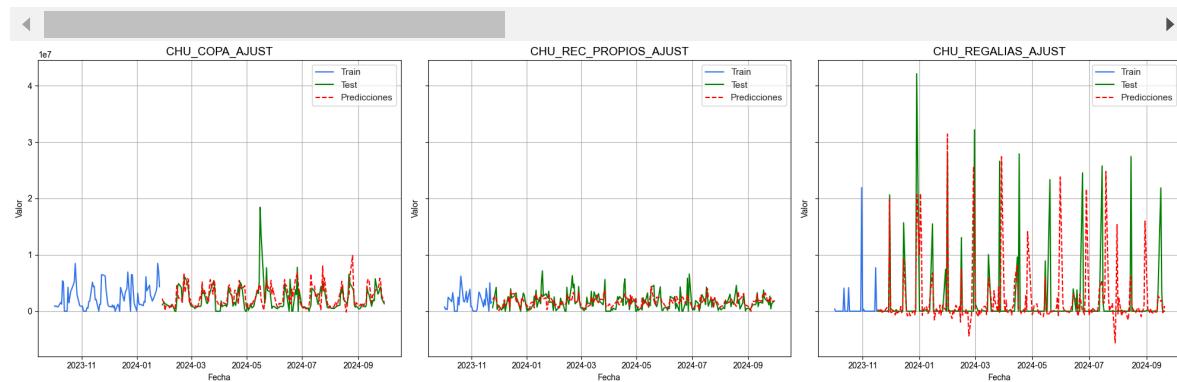
```

```
In [6]: pd.options.display.float_format = '{:.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=[pd.Series(dataframes_train[i][titulos[i]], index=dataframe
    dataframes_test=[pd.Series(dataframes_test[i][titulos[i]], index=dataframes_
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2023-10-01'
)))

```

	model	name	len_train	l
0	LGBMRegressor(colsample_bytree=0.8076995248364...	CHU_COPA_AJUST	1584	
1	LGBMRegressor(colsample_bytree=0.5125884353625...	CHU_REC_PROPIOS_AJUST	1995	
2	LGBMRegressor(colsample_bytree=0.8651927912102...	CHU_REGALIAS_AJUST	1985	



None

```
In [ ]:
```

LIGHTGBM

TRAIN 90%

TEST 10%

AUTOTS

TRAIN 90%

TEST 10%

```
In [1]: # Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from autots import AutoTS
import matplotlib.pyplot as plt
import funciones
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
```

```
In [2]: df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-")
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPIA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0], end=dataframes[i].index[-1]))
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPIA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

```
In [3]: # TRAIN TEST
n_train = 0.9
train_copa = dataframes[0].iloc[:round(len(dataframes[0])*n_train)]
test_copa = dataframes[0].iloc[round(len(dataframes[0])*n_train):]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[:round(len(dataframes[1])*n_train)]
test_recursos = dataframes[1].iloc[round(len(dataframes[1])*n_train):]
print(f"Recursos: train({train_recursos.shape}), test({test_recursos.shape})")

train_regalias = dataframes[2].iloc[:round(len(dataframes[2])*n_train)]
test_regalias = dataframes[2].iloc[round(len(dataframes[2])*n_train):]
print(f"Regalias: train({train_regalias.shape}), test({test_regalias.shape})")

dataframes_train = [ train_copa, train_recursos, train_regalias ]
dataframes_test = [ test_copa, test_recursos, test_regalias ]
```

```
Coparticipacion: train((1584,)), test((176,))
Recursos: train((1995,)), test((222,))
Regalias: train((1985,)), test((221,))
```

```
In [ ]: results_train_test = []
predictions_test = []
for i, df in enumerate(dataframes_train):
    df_train = df
    df_test = dataframes_test[i]

    model = AutoTS(
        forecast_length=len(dataframes_test[i]),
        frequency="B",
        prediction_interval=0.95,
        ensemble=None,
        models_mode='deep',
        model_list = 'superfast',
```

```

    max_generations=10,    # intenta optimizar el modelo a traves de 10 iteraciones
    num_validations=3,
    no_negatives=True,
    n_jobs='auto')
modelAutoTS = model.fit(df_train)
# Find the best parameters

fechas = pd.date_range(start=df_test.index.min(), end=df_test.index.max(), freq='H')
pred_test = model.predict(forecast_length=len(fechas)).forecast
predictions_test.append(pred_test)
# Cálculo del MSE en el conjunto de prueba
mape_test = mean_absolute_percentage_error(df_test, pred_test)
mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(df_test))
mse_test = mean_squared_error(df_test, pred_test)
mae_test = mean_absolute_error(df_test, pred_test)
rmse = np.sqrt(mean_squared_error(df_test, pred_test))
results_train_test.append({
    "model": modelAutoTS,
    "name": df_train.name,
    "len_train": len(df_train),
    "len_test": len(df_test),
    "mape_test": mape_test,
    "mse_test": mse_test,
    "mape_mean": mape_mean,
    "mae_test": mae_test,
    "rmse": rmse
})

```

In [5]:

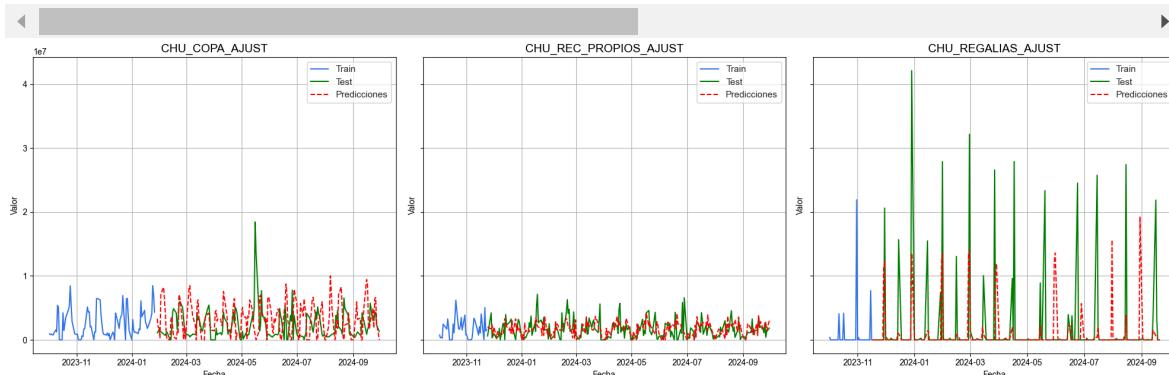
```

pd.options.display.float_format = '{:.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=dataframes_train,
    dataframes_test=dataframes_test,
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2023-10-01'
))

```

model		name	len_train	len_test	mape_test
0	Initiated AutoTS object with best model: \nSea...	CHU_COPA_AJUST	1584	176	1,069,270,913,042,505,662,464.00 1
1	Initiated AutoTS object with best model: \nGLS...	CHU_REC_PROPIOS_AJUST	1995	222	476,620,172,305,521,311,744.00
2	Initiated AutoTS object with best model: \nSea...	CHU_REGALIAS_AJUST	1985	221	2,174,434,595,027,651,395,584.00 3



None

```
In [6]: results = pd.DataFrame(results_train_test)
```

```
for i, row in results.iterrows():
    display(row["name"])
    print(row.model)
```

```
'CHU_COPA_AJUST'
Initiated AutoTS object with best model:
SeasonalNaive
{'fillna': 'pchip', 'transformations': {'0': 'SeasonalDifference', '1': 'MaxAbsScaler'}, 'transformation_params': {'0': {'lag_1': 7, 'method': 'LastValue'}, '1': {}}}
{'method': 'lastvalue', 'lag_1': 96, 'lag_2': 28}
Validation: 0, 1, 2, 3
SMAPE: nan, nan, nan, nan
MAE: 2822593.394881551, 2705886.889956714, 2266922.725699413, 2126160.779481562
SPL: 0.34733674670580267, 0.33363305331363236, 0.3567839576534018, 0.398042596097
04773
'CHU_REC_PROPIOS_AJUST'
```

```
Initiated AutoTS object with best model:  
GLS  
{'fillna': 'ffill', 'transformations': {'0': 'DatepartRegression', '1': 'Historic  
Values', '2': 'SeasonalDifference'}, 'transformation_params': {'0': {'regression_  
model': {'model': 'ElasticNet', 'model_params': {'l1_ratio': 0.1, 'fit_intercep  
t': True, 'selection': 'cyclic'}}, 'datepart_method': 'simple_2', 'polynomial_deg  
ree': 2, 'transform_dict': {'fillna': None, 'transformations': {'0': 'ClipOutlier  
s'}}, 'transformation_params': {'0': {'method': 'clip', 'std_threshold': 4}}}, 'ho  
liday_countries_used': False, 'lags': None, 'forward_lags': None}, '1': {'windo  
w': 730}, '2': {'lag_1': 24, 'method': 2}}}  
{}  
Validation: 0, 1, 2, 3  
SMAPE: nan, nan, nan, nan  
MAE: 1226828.4737256807, 958627.634529741, 872269.9874219908, 777486.4746628217  
SPL: 0.2207936109057398, 0.20243127261189903, 0.1972549963009233, 0.1784455234438  
5997  
'CHU_REGALIAS_AJUST'  
Initiated AutoTS object with best model:  
SeasonalityMotif  
{'fillna': 'pchip', 'transformations': {'0': 'PositiveShift', '1': 'HistoricValue  
s'}, 'transformation_params': {'0': {}, '1': {'window': None}}}  
{'window': 10, 'point_method': 'midhinge', 'distance_metric': 'minkowski', 'k': 1  
0, 'datepart_method': 'recurring', 'independent': True}  
Validation: 0, 1, 2, 3  
SMAPE: nan, nan, nan, nan  
MAE: 1399081.8144796381, 1430369.8235294118, 1441480.6425339365, 1386704.18099547  
5  
SPL: 0.13252659123779703, 0.11893795878406024, 0.09366669094475673, 0.20322019724  
881865
```

In []:

PROPHET

TEST 30 DÍAS

In [1]:

```
# Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
import matplotlib.pyplot as plt
from prophet.plot import plot_cross_validation_metric
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
import funciones
```

In [2]:

```
df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-"
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0],
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

In [3]:

```
# TRAIN TEST
n_test = 30
train_copa = dataframes[0].iloc[:-n_test]
test_copa = dataframes[0].iloc[-n_test:]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[:-n_test]
test_recursos = dataframes[1].iloc[-n_test:]
print(f"Recursos: train({train_recursos.shape}), test({test_recursos.shape})")

train_regalias = dataframes[2].iloc[:-n_test]
test_regalias = dataframes[2].iloc[-n_test:]
print(f"Regalias: train({train_regalias.shape}), test({test_regalias.shape})")

dataframes_train = [train_copa, train_recursos, train_regalias]
dataframes_test = [test_copa, test_recursos, test_regalias]
```

Coparticipacion: train((1730,)), test((30,))

Recursos: train((2187,)), test((30,))

Regalias: train((2176,)), test((30,))

In [4]:

```
results_train_test = []
predictions_test = []
from prophet.make_holidays import make_holidays_df
best_params = pd.read_csv("prophet_best_params.csv")
best_fourier = pd.read_csv("best_fourier_orders.csv")

for i, df_train in enumerate(dataframes_train):
    df_train = df_train.to_frame()
    df_train.reset_index(inplace=True)
    df_train.columns = ["ds", "y"]
```

```

df_test = dataframes_test[i]
params = eval(best_params.iloc[i]["best_params"])
year_list = dataframes[i].index.year.unique()
holidays = make_holidays_df(year_list=year_list, country='AR')
model = Prophet(**params,holidays=holidays)

fourier_yearly = eval(best_fourier.iloc[i]["Fourier_yearly"])["fourier_order"]
fourier_monthly = eval(best_fourier.iloc[i]["Fourier_monthly"])["fourier_order"]
model.add_seasonality(name='monthly', period=30.5, fourier_order=fourier_monthly)
model.add_seasonality(name='yearly', period=365.25, fourier_order=fourier_yearly)

model.fit(df_train)
fechas = pd.date_range(start=df_test.index.min(), end=df_test.index.max(), freq='B')
future = model.make_future_dataframe(periods=len(fechas), freq='B')
pred_test = model.predict(future)
pred_test.index = pred_test["ds"]
pred_test = pred_test["yhat"]
pred_test = pred_test[-len(df_test):]

predictions_test.append(pred_test)
# Cálculo del MSE en el conjunto de prueba
mape_test = mean_absolute_percentage_error(df_test, pred_test)
mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(df_test))
mse_test = mean_squared_error(df_test, pred_test)
mae_test = mean_absolute_error(df_test, pred_test)
rmse = np.sqrt(mean_squared_error(df_test, pred_test))
results_train_test.append({
    "model": model,
    "name": titulos[i],
    "len_train": len(df_train),
    "len_test": len(df_test),
    "mape_test": mape_test,
    "mse_test": mse_test,
    "mape_mean": mape_mean,
    "mae_test": mae_test,
    "rmse": rmse
})

```

```

13:00:15 - cmdstanpy - INFO - Chain [1] start processing
13:00:15 - cmdstanpy - INFO - Chain [1] done processing
13:00:16 - cmdstanpy - INFO - Chain [1] start processing
13:00:16 - cmdstanpy - INFO - Chain [1] done processing
13:00:17 - cmdstanpy - INFO - Chain [1] start processing
13:00:17 - cmdstanpy - INFO - Chain [1] done processing

```

In [5]:

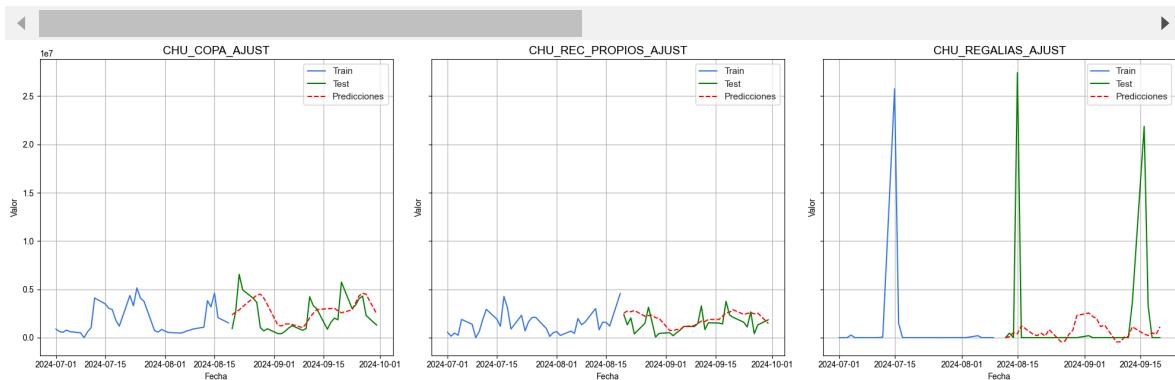
```

pd.options.display.float_format = '{:.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=dataframes_train,
    dataframes_test=dataframes_test,
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2024-07-01'
))

```

model	name	len_train	len_test
0 <prophet.forecaster.Prophet object at 0x000002...	CHU_COPA_AJUST	1730	30
1 <prophet.forecaster.Prophet object at 0x000002...	CHU_REC_PROPIOS_AJUST	2187	30
2 <prophet.forecaster.Prophet object at 0x000002...	CHU_REGALIAS_AJUST	2176	30 2,422,839,158,73



None

In []:

XBOOST

TEST 30 DÍAS

In [1]:

```
# Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
import matplotlib.pyplot as plt
from prophet.plot import plot_cross_validation_metric
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
import funciones
import xgboost as xgb
```

In [2]:

```
df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-"
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0],
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

In [3]:

```
def extract_time_features(index):
    return pd.Series({
        'dayofweek': index.dayofweek,
        'quarter': index.quarter,
        'month': index.month,
        'year': index.year,
        'dayofyear': index.dayofyear,
        'dayofmonth': index.day,
        'weekofyear': index.isocalendar().week
    })
def add_lags(df, titulo):
    target_map = df[titulo].to_dict()
    df['lag1'] = (df.index - pd.Timedelta('364 days')).map(target_map) # df_1['P'
    df['lag2'] = (df.index - pd.Timedelta('728 days')).map(target_map) # df_1['P'
    df['lag3'] = (df.index - pd.Timedelta('1092 days')).map(target_map) # df_1['P'
    return df

for i in range(len(dataframes)):
    time_features = dataframes[i].index.to_series().apply(extract_time_features)
    dataframes[i] = pd.concat([dataframes[i], time_features], axis=1)
    dataframes[i] = add_lags(dataframes[i], titulos[i])
```

In [4]:

```
# TRAIN TEST
n_test = 30
train_copa = dataframes[0].iloc[:n_test]
test_copa = dataframes[0].iloc[-n_test:]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")
```

```

train_recursos = dataframes[1].iloc[:n_test]
test_recursos = dataframes[1].iloc[-n_test:]
print(f'Recursos: train({train_recursos.shape}), test({test_recursos.shape})')

train_regalias = dataframes[2].iloc[:n_test]
test_regalias = dataframes[2].iloc[-n_test:]
print(f'Regalias: train({train_regalias.shape}), test({test_regalias.shape})')

dataframes_train = [ train_copa, train_recursos, train_regalias ]
dataframes_test = [ test_copa, test_recursos, test_regalias ]

```

Coparticipacion: train((1730, 11)), test((30, 11))
 Recursos: train((2187, 11)), test((30, 11))
 Regalias: train((2176, 11)), test((30, 11))

```

In [ ]: results_train_test = []
predictions_test = []
best_params = pd.read_csv("xgb_best_params.csv", sep=";")

for i, df_train in enumerate(dataframes_train):

    params = eval(best_params.iloc[i]["best_params"])

    FEATURES = ['dayofweek', 'quarter', 'month', 'year', 'dayofyear' , 'dayofmon'
    TARGET = titulos[i]

    df_test = dataframes_test[i][TARGET]
    model = xgb.XGBRegressor(**params )

    X_train = dataframes_train[i][FEATURES]
    y_train = dataframes_train[i][TARGET]
    X_test= dataframes_test[i][FEATURES]
    y_test= dataframes_test[i][TARGET]

    model.fit(
        X_train, y_train,
        eval_set=[(X_test, y_test)])
    )

    pred_test = model.predict(X_test)
    pred_test = pd.Series(pred_test, index=dataframes_test[i].index)
    predictions_test.append(pred_test)

    # Cálculo del MSE en el conjunto de prueba
    mape_test = mean_absolute_percentage_error(df_test, pred_test)
    mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(df_test))
    mse_test = mean_squared_error(df_test, pred_test)
    mae_test = mean_absolute_error(df_test, pred_test)
    rmse = np.sqrt(mean_squared_error(df_test, pred_test))
    results_train_test.append({
        "model": model,
        "name": titulos[i],
        "len_train": len(df_train),
        "len_test": len(df_test),
        "mape_test": mape_test,
        "mse_test":mse_test,
        "mape_mean": mape_mean,
        "mae_test": mae_test,
        "rmse": rmse
    })

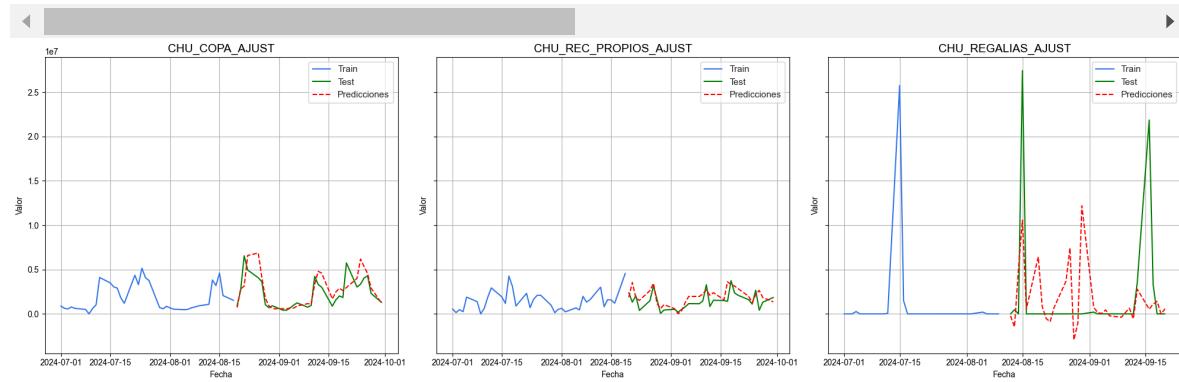
```

```
In [6]: pd.options.display.float_format = '{:,.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=[pd.Series(dataframes_train[i][titulos[i]], index=dataframe
    dataframes_test=[pd.Series(dataframes_test[i][titulos[i]], index=dataframes_
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2024-07-01'
)))

```

	model	name	len_train	len_test
0	XGBRegressor(base_score=None, booster=None, ca...	CHU_COPA_AJUST	1730	30
1	XGBRegressor(base_score=None, booster=None, ca...	CHU_REC_PROPIOS_AJUST	2187	30
2	XGBRegressor(base_score=None, booster=None, ca...	CHU_REGALIAS_AJUST	2176	30 5,880,061,71



None

```
In [ ]:
```

LIGHTGBM

TEST 30 DÍAS

```
In [1]: # Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from prophet import Prophet
from prophet.diagnostics import cross_validation
from prophet.diagnostics import performance_metrics
import matplotlib.pyplot as plt
from prophet.plot import plot_cross_validation_metric
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
import funciones
```

```
In [2]: df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-"
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0],
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

```
In [3]: def extract_time_features(index):
    return pd.Series({
        'dayofweek': index.dayofweek,
        'quarter': index.quarter,
        'month': index.month,
        'year': index.year,
        'dayofyear': index.dayofyear,
        'dayofmonth': index.day,
        'weekofyear': index.isocalendar().week
    })
def add_lags(df, titulo):
    target_map = df[titulo].to_dict()
    df['lag1'] = (df.index - pd.Timedelta('364 days')).map(target_map) # df_1['P']
    df['lag2'] = (df.index - pd.Timedelta('728 days')).map(target_map) # df_1['P']
    df['lag3'] = (df.index - pd.Timedelta('1092 days')).map(target_map) # df_1['P'
    return df

for i in range(len(dataframes)):
    time_features = dataframes[i].index.to_series().apply(extract_time_features)
    dataframes[i] = pd.concat([dataframes[i], time_features], axis=1)
    dataframes[i] = add_lags(dataframes[i], titulos[i])
```

```
In [4]: # TRAIN TEST
n_test = 30
train_copa = dataframes[0].iloc[:n_test]
test_copa = dataframes[0].iloc[-n_test:]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[:n_test]
```

```

test_recursos = dataframes[1].iloc[-n_test:]
print(f'Recursos: train({train_recursos.shape}), test({test_recursos.shape})')

train_Regalias = dataframes[2].iloc[:n_test]
test_Regalias = dataframes[2].iloc[-n_test:]
print(f'Regalias: train({train_Regalias.shape}), test({test_Regalias.shape})')

dataframes_train = [ train_copa, train_recursos, train_Regalias ]
dataframes_test = [ test_copa, test_recursos, test_Regalias ]

```

Coparticipacion: train((1730, 11)), test((30, 11))
 Recursos: train((2187, 11)), test((30, 11))
 Regalias: train((2176, 11)), test((30, 11))

```

In [ ]: results_train_test = []
predictions_test = []
best_params = pd.read_csv("lgbm_best_params.csv", sep=";")
import lightgbm as lgb

for i, df_train in enumerate(dataframes_train):

    params = eval(best_params.iloc[i]["best_params"])

    FEATURES = ['dayofweek', 'quarter', 'month', 'year', 'dayofyear' , 'dayofmon'
    TARGET = titulos[i]

    df_test = dataframes_test[i][TARGET]
    model = lgb.LGBMRegressor(**params )

    X_train = dataframes_train[i][FEATURES]
    y_train = dataframes_train[i][TARGET]
    X_test= dataframes_test[i][FEATURES]
    y_test= dataframes_test[i][TARGET]

    model.fit(df_train[FEATURES], df_train[TARGET],
              eval_set=[(X_train, y_train), (X_test, y_test)], eval_metric="rmse"

    pred_test = model.predict(dataframes_test[i][FEATURES])
    pred_test = pd.Series(pred_test, index=dataframes_test[i].index)
    predictions_test.append(pred_test)

    # Cálculo del MSE en el conjunto de prueba
    mape_test = mean_absolute_percentage_error(df_test, pred_test)
    mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(df_test))
    mse_test = mean_squared_error(df_test, pred_test)
    mae_test = mean_absolute_error(df_test, pred_test)
    rmse = np.sqrt(mean_squared_error(df_test, pred_test))
    results_train_test.append({
        "model": model,
        "name": titulos[i],
        "len_train": len(df_train),
        "len_test": len(df_test),
        "mape_test": mape_test,
        "mse_test": mse_test,
        "mape_mean": mape_mean,
        "mae_test": mae_test,
        "rmse": rmse
    })

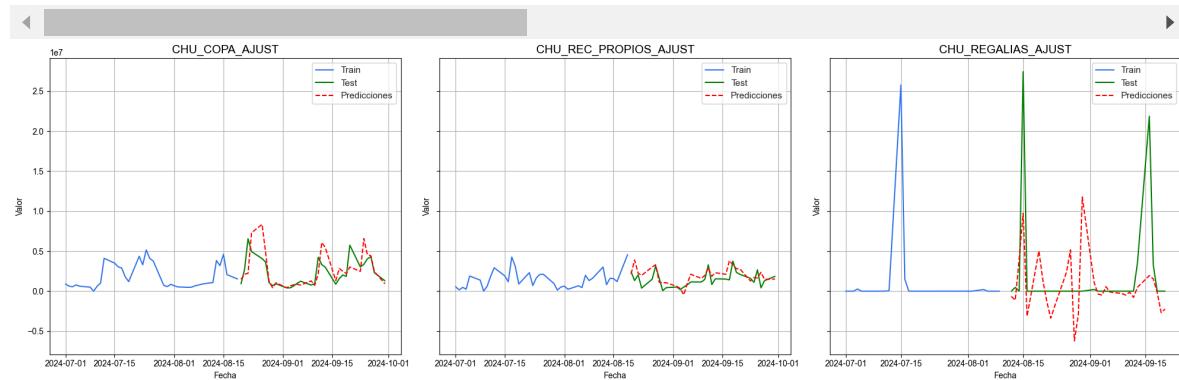
```

```
In [6]: pd.options.display.float_format = '{:.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=[pd.Series(dataframes_train[i][titulos[i]], index=dataframe
    dataframes_test=[pd.Series(dataframes_test[i][titulos[i]], index=dataframes_
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2024-07-01'
)))

```

	model	name	len_train	l
0	LGBMRegressor(colsample_bytree=0.8076995248364...	CHU_COPA_AJUST	1730	
1	LGBMRegressor(colsample_bytree=0.5125884353625...	CHU_REC_PROPIOS_AJUST	2187	
2	LGBMRegressor(colsample_bytree=0.8651927912102...	CHU_REGALIAS_AJUST	2176	



None

```
In [ ]:
```

AUTOTS

TEST 30 DÍAS

```
In [1]: # Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from autots import AutoTS
import matplotlib.pyplot as plt
import funciones
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
```

```
In [2]: df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-")
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPIA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0],
    end=dataframes[i].index[-1]))
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPIA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

```
In [3]: # TRAIN TEST
n_test = 30
train_copa = dataframes[0].iloc[:n_test]
test_copa = dataframes[0].iloc[-n_test:]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[:n_test]
test_recursos = dataframes[1].iloc[-n_test:]
print(f"Recursos: train({train_recursos.shape}), test({test_recursos.shape})")

train_regalias = dataframes[2].iloc[:n_test]
test_regalias = dataframes[2].iloc[-n_test:]
print(f"Regalias: train({train_regalias.shape}), test({test_regalias.shape})")

dataframes_train = [ train_copa, train_recursos, train_regalias ]
dataframes_test = [ test_copa, test_recursos, test_regalias ]
```

Coparticipacion: train((1730,)), test((30,))
Recursos: train((2187,)), test((30,))
Regalias: train((2176,)), test((30,))

```
In [ ]: results_train_test = []
predictions_test = []
for i, df in enumerate(dataframes_train):
    df_train = df
    df_test = dataframes_test[i]

    model = AutoTS(
        forecast_length=len(dataframes_test[i]),
        frequency="B",
        prediction_interval=0.95,
        ensemble=None,
        models_mode='deep',
        model_list = 'superfast',
```

```

    max_generations=10,    # intenta optimizar el modelo a traves de 10 iteraciones
    num_validations=3,
    no_negatives=True,
    n_jobs='auto')
modelAutoTS = model.fit(df_train)
# Find the best parameters

fechas = pd.date_range(start=df_test.index.min(), end=df_test.index.max(), freq='H')
pred_test = model.predict(forecast_length=len(fechas)).forecast
predictions_test.append(pred_test)
# Cálculo del MSE en el conjunto de prueba
mape_test = mean_absolute_percentage_error(df_test, pred_test)
mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(df_test))
mse_test = mean_squared_error(df_test, pred_test)
mae_test = mean_absolute_error(df_test, pred_test)
rmse = np.sqrt(mean_squared_error(df_test, pred_test))
results_train_test.append({
    "model": modelAutoTS,
    "name": df_train.name,
    "len_train": len(df_train),
    "len_test": len(df_test),
    "mape_test": mape_test,
    "mse_test": mse_test,
    "mape_mean": mape_mean,
    "mae_test": mae_test,
    "rmse": rmse
})

```

In [7]:

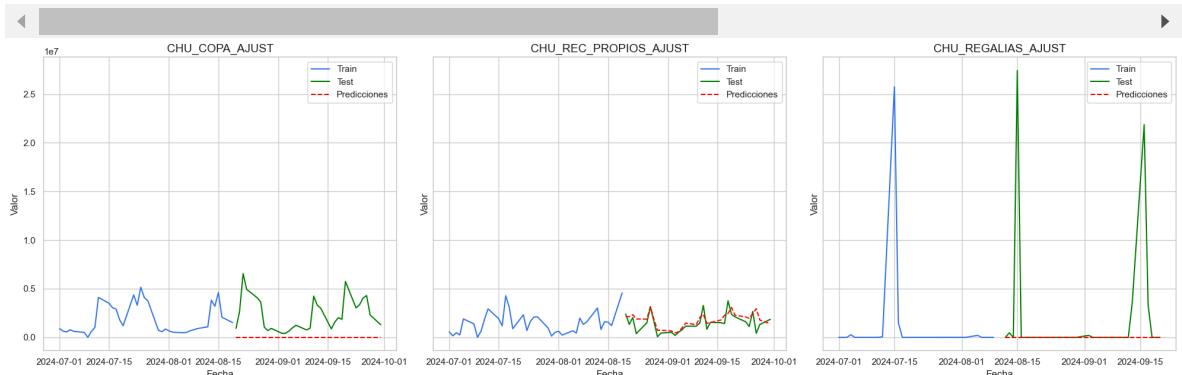
```

pd.options.display.float_format = '{:.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=dataframes_train,
    dataframes_test=dataframes_test,
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2024-07-01'
))

```

model		name	len_train	len_test	mape_test	mse_test
0	Initiated AutoTS object with best model: \nLas...	CHU_COPA_AJUST	1730	30	1.00	8,545,162,889,493.53
1	Initiated AutoTS object with best model: \nSea...	CHU_REC_PROPIOS_AJUST	2187	30	0.99	547,894,342,942.74
2	Initiated AutoTS object with best model: \nSea...	CHU_REGALIAS_AJUST	2176	30	0.27	41,804,929,164,773.93



None

```
In [6]: results = pd.DataFrame(results_train_test)

for i, row in results.iterrows():
    display(row["name"])
    print(row.model)

'CHU_COPA_AJUST'
```

```
Initiated AutoTS object with best model:  
LastValueNaive  
{'fillna': 'quadratic', 'transformations': {'0': 'RollingMeanTransformer', '1': 'AnomalyRemoval', '2': 'bkfilter', '3': 'ScipyFilter'}, 'transformation_params': {'0': {'fixed': True, 'window': 360, 'macro_micro': True, 'center': True}, '1': {'method': 'zscore', 'method_params': {'distribution': 'uniform', 'alpha': 0.05}, 'fillna': 'ffill', 'transform_dict': {'fillna': None, 'transformations': {'0': 'EWMAFilter'}}, 'transformation_params': {'0': {'span': 7}}}, 'isolated_only': False}, '2': {}, '3': {'method': 'butter', 'method_args': {'N': 2, 'btype': 'highpass', 'analog': False, 'output': 'sos', 'Wn': 0.015384615384615385}}}}}  
{}  
Validation: 0, 1, 2, 3  
SMAPE: 80.45022147190438, nan, nan, nan  
MAE: 1457881.601770488, 1754848.8054795982, 3351528.787366555, 1944707.5176069036  
SPL: 0.18310796907270413, 0.3516077000937226, 1.516704946141438, 0.23639689726808  
616  
'CHU_REC_PROPIOS_AJUST'  
Initiated AutoTS object with best model:  
SeasonalityMotif  
{'fillna': 'akima', 'transformations': {'0': 'PositiveShift', '1': 'Detrend'}, 'transformation_params': {'0': {}, '1': {'model': 'GLS', 'phi': 1, 'window': 365, 'transform_dict': None}}}  
{'window': 15, 'point_method': 'midhinge', 'distance_metric': 'mae', 'k': 20, 'depart_method': 'simple', 'independent': False}  
Validation: 0, 1, 2, 3  
SMAPE: 46.63511981455654, 81.92051984679205, 67.9873722891953, 59.817692083104504  
MAE: 672472.5785951081, 1013335.51811632, 1024976.4424628529, 665533.8818280211  
SPL: 0.14172371881288814, 0.22842532323077214, 0.16493539716519867, 0.16548226664  
122395  
'CHU_REGALIAS_AJUST'  
Initiated AutoTS object with best model:  
SeasonalityMotif  
{'fillna': 'ffill', 'transformations': {'0': 'CenterLastValue', '1': 'QuantileTransformer'}, 'transformation_params': {'0': {'rows': 6}, '1': {'output_distribution': 'uniform', 'n_quantiles': 1000}}}  
{'window': 15, 'point_method': 'midhinge', 'distance_metric': 'mse', 'k': 20, 'depart_method': 'simple', 'independent': True}  
Validation: 0, 1, 2, 3  
SMAPE: nan, nan, nan, nan  
MAE: 926159.4666666667, 1861666.2666666666, 1564803.1333333333, 2344361.2666666666  
6  
SPL: 0.19809204672333816, 0.548411223643126, 0.21565462335433694, 0.159545677128  
9861
```

In []:

TIME-GPT

TEST 30 DÍAS

```
In [1]: # Python
import itertools
import numpy as np
import pandas as pd
import pandas as pd
import numpy as np
from autots import AutoTS
import matplotlib.pyplot as plt
import funciones
from sklearn.metrics import mean_squared_error, mean_absolute_percentage_error,
```

```
In [2]: df_main = pd.read_excel("https://raw.githubusercontent.com/carrenogf/MCD-Series-")
df_main = df_main.sort_values("FECHA", ascending=True)
df_main.set_index("FECHA", inplace=True)
df_copa = df_main["CHU_COPIA_AJUST"].dropna()
df_recprop = df_main["CHU_REC_PROPIOS_AJUST"].dropna()
df_regal = df_main["CHU_REGALIAS_AJUST"].dropna()
dataframes = [df_copa, df_recprop, df_regal]
for i in range(len(dataframes)):
    dataframes[i] = dataframes[i].reindex(pd.date_range(start=dataframes[i].index[0],
    end=dataframes[i].index[-1]))
    dataframes[i] = dataframes[i].fillna(0)

titulos = ["CHU_COPIA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

```
In [3]: # TRAIN TEST
n_test = 30
train_copa = dataframes[0].iloc[: - n_test]
test_copa = dataframes[0].iloc[- n_test:]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[: - n_test]
test_recursos = dataframes[1].iloc[- n_test:]
print(f"Recursos: train({train_recursos.shape}), test({test_recursos.shape})")

train_regalias = dataframes[2].iloc[: - n_test]
test_regalias = dataframes[2].iloc[- n_test:]
print(f"Regalias: train({train_regalias.shape}), test({test_regalias.shape})")

dataframes_train = [train_copa, train_recursos, train_regalias]
dataframes_test = [test_copa, test_recursos, test_regalias]
```

```
Coparticipacion: train((1730,)), test((30,))
Recursos: train((2187,)), test((30,))
Regalias: train((2176,)), test((30,))
```

```
In [4]: from nixtla import NixtlaClient
import warnings
warnings.filterwarnings("ignore")
pd.options.display.float_format = '{:.2f}'.format
nixtla_client = NixtlaClient(
    # defaults to os.environ.get("NIXTLA_API_KEY")
    api_key = 'nixak-P7309LQjFeXHr6GYXQ1H41AsZEadk15X66HvvheepbODf11yrBGLBt6bPxX'
)
```

```
In [ ]: results_train_test = []
predictions_test = []
forecast = []
for i, df in enumerate(dataframes_train):
```

```

name = df.name
df_train = df.to_frame()
df_train = df_train.reset_index()
df_test = dataframes_test[i]
fechas = pd.date_range(start=df_test.index.min(), end=df_test.index.max(), f

response = nixtla_client.forecast(
    df=df_train, # Entrenamiento solo con los datos de train
    time_col="index", # Nombre del índice de tiempo (columna de fecha)
    target_col=name, # Columna objetivo
    freq="B", # Frecuencia de los datos (diaria)
    h=len(fechas), # El tamaño del conjunto de test
    level=[80], # Niveles de confianza
    finetune_loss='rmse',
    finetune_steps=20,
)
# Convertir la respuesta a DataFrame
forecast_df = pd.DataFrame(response)
forecast.append(forecast_df)
# Find the best parameters

pred_test = forecast_df["TimeGPT"].values
pred_test = pd.Series(pred_test, index=df_test.index)
predictions_test.append(pred_test)
# Cálculo del MSE en el conjunto de prueba
mape_test = mean_absolute_percentage_error(df_test, pred_test)
mape_mean = mean_absolute_percentage_error(df_test, [df_test.mean()] * len(df_test))
mse_test = mean_squared_error(df_test, pred_test)
mae_test = mean_absolute_error(df_test, pred_test)
rmse = np.sqrt(mean_squared_error(df_test, pred_test))
results_train_test.append({
    "model": "TimeGPT",
    "name": name,
    "len_train": len(df_train),
    "len_test": len(df_test),
    "mape_test": mape_test,
    "mse_test": mse_test,
    "mape_mean": mape_mean,
    "mae_test": mae_test,
    "rmse": rmse
})

```

In [6]:

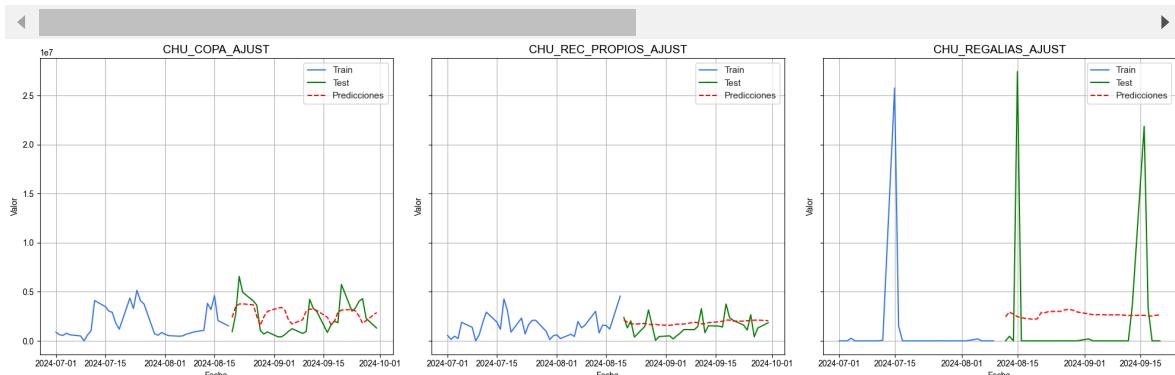
```

pd.options.display.float_format = '{:.2f}'.format
display(pd.DataFrame(results_train_test))

display(funciones.plot_train_test_predictions(
    dataframes_train=dataframes_train,
    dataframes_test=dataframes_test,
    predictions_test=predictions_test,
    series_names=titulos,
    start_date='2024-07-01'
))

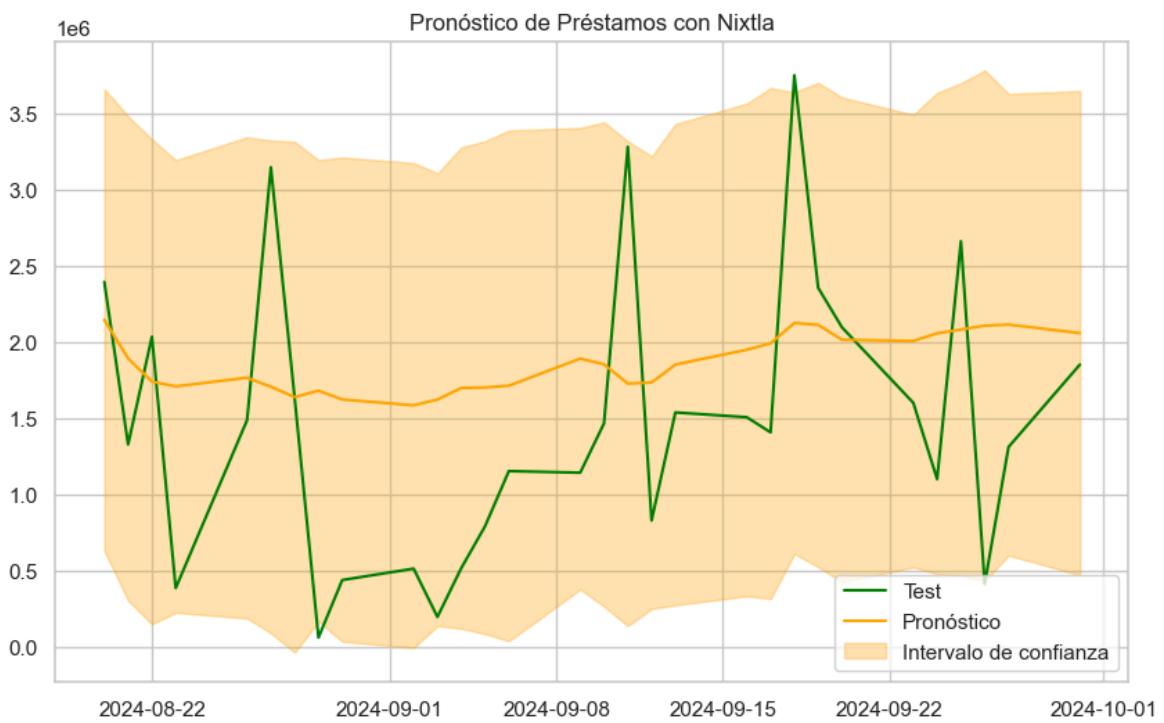
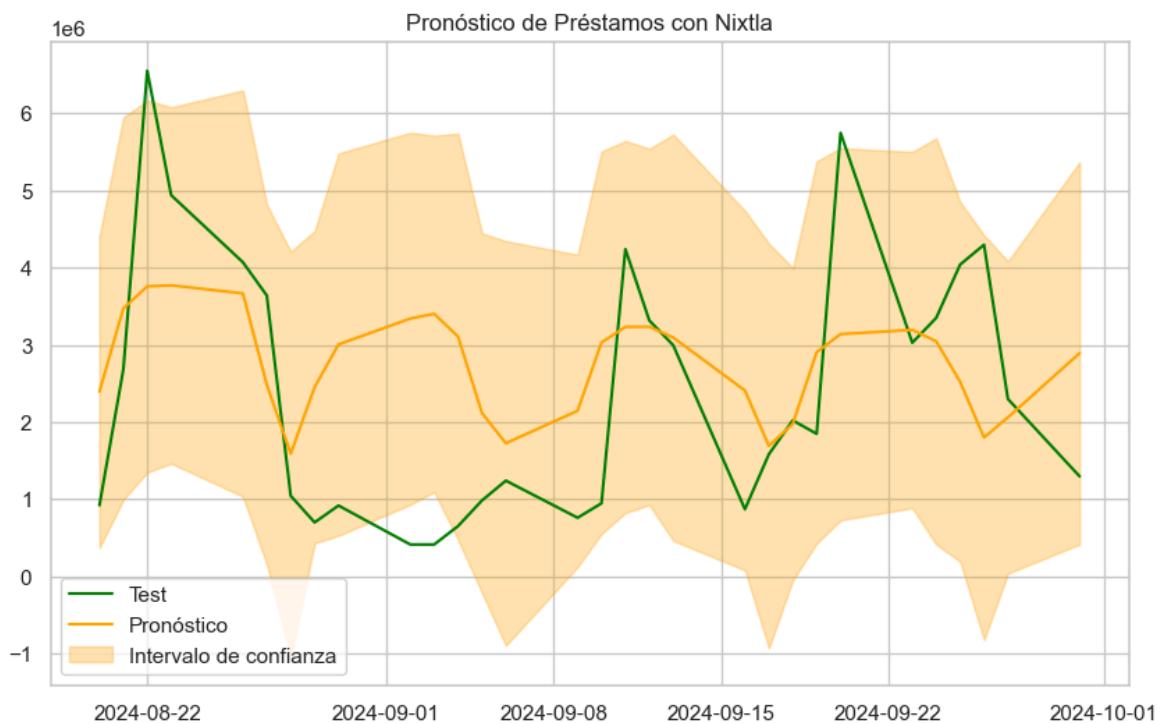
```

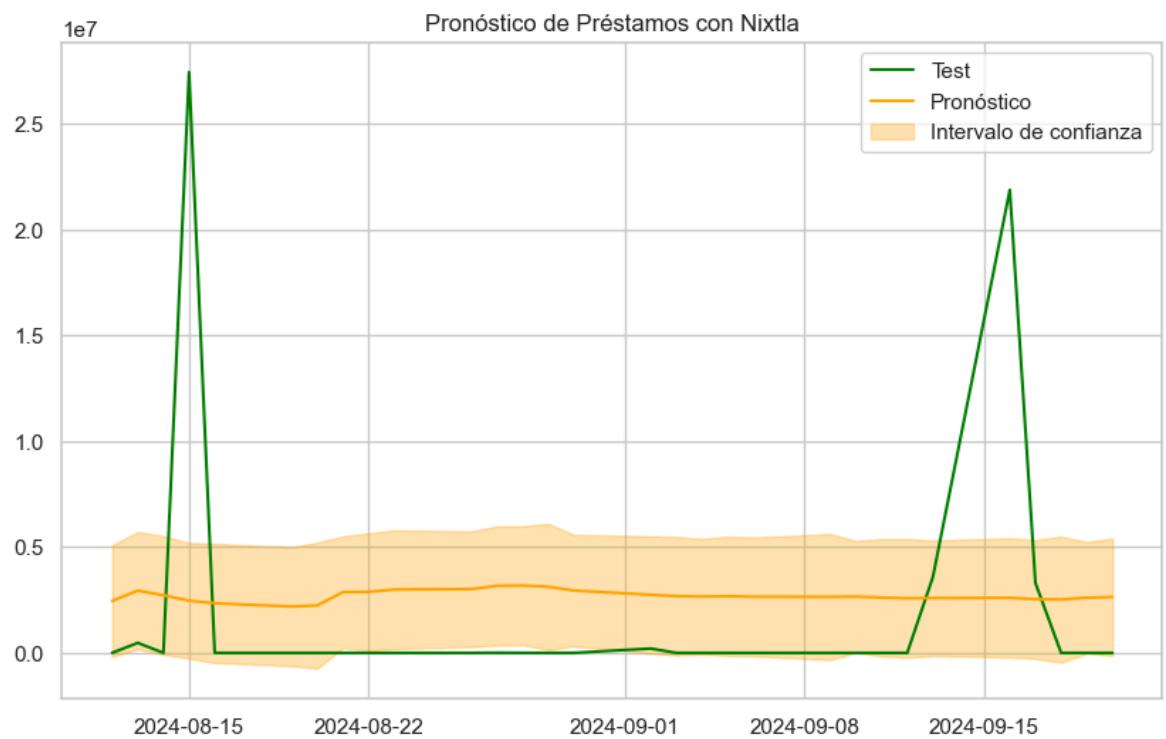
model	name	len_train	len_test	mape_test
0 TimeGPT	CHU_COPA_AJUST	1730	30	1.25
1 TimeGPT	CHU_REC_PROPIOS_AJUST	2187	30	1.90
2 TimeGPT	CHU_REGALIAS_AJUST	2176	30	8,908,123,590,758,549,684,224.00



None

```
In [66]: import matplotlib.pyplot as plt
def plot_pred(i):
    plt.figure(figsize=(10, 6))
    #plt.plot(train.index, train['CHU_COPA_AJUST'], label='Train', color='blue')
    plt.plot(dataframes_test[i].index, dataframes_test[i], label='Test', color='green')
    plt.plot(predictions_test[i].index, predictions_test[i], label='Pronóstico', color='red')
    plt.fill_between(
        forecast[i]["index"],
        forecast[i]["TimeGPT-lo-80"],
        forecast[i]["TimeGPT-hi-80"],
        color='orange',
        alpha=0.3,
        label='Intervalo de confianza'
    )
    plt.legend()
    plt.title("Pronóstico de Préstamos con Nixtla")
    plt.show()
for i in range(len(dataframes_test)):
    plot_pred(i)
```





HIPER PARAMETROS OPTIMIZADOS

PROPHET

df	best_params
CHU_COPA_AJUST	{'changepoint_prior_scale': 0.1, 'seasonality_prior_scale': 0.01, 'daily_seasonality': True, 'yearly_seasonality': True, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'changepoint_range': 0.8}
CHU_REC_PROPIOS_AJUST	{'changepoint_prior_scale': 0.001, 'seasonality_prior_scale': 10.0, 'daily_seasonality': True, 'yearly_seasonality': True, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'changepoint_range': 0.9}
CHU_REGALIAS_AJUST	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 0.1, 'daily_seasonality': True, 'yearly_seasonality': False, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'changepoint_range': 0.8}

PROPHET FOURIER ORDER

df	Fourier_yearly	Fourier_monthly
CHU_COPA_AJUST	{'fourier_order': 5, 'rmse': 2114568.5994672766}	{'fourier_order': 3, 'rmse': 1829278.0979960766}
CHU_REC_PROPIOS_AJUST	{'fourier_order': 10, 'rmse': 1241865.2497155124}	{'fourier_order': 3, 'rmse': 1160714.3127307887}
CHU_REGALIAS_AJUST	{'fourier_order': 3, 'rmse': 6054561.279307909}	{'fourier_order': 3, 'rmse': 5786977.886510033}

XGBOOST

df	best_params
CHU_COPA_AJUST	{'n_estimators': 453, 'learning_rate': 0.052251207169970344, 'max_depth': 3, 'min_child_weight': 2, 'subsample': 0.9052659212939362, 'colsample_bytree': 0.9700609066338445, 'gamma': 0.4972981188401708, 'reg_alpha': 0.49902287558195313, 'reg_lambda': 1.390212005740913}
CHU_REC_PROPIOS_AJUST	{'n_estimators': 370, 'learning_rate': 0.07063010235445233, 'max_depth': 7, 'min_child_weight': 4, 'subsample': 0.6846124207200684, 'colsample_bytree': 0.7520959246442501, 'gamma': 0.34727747835872463, 'reg_alpha': 0.10152547979284736, 'reg_lambda': 1.4618068656153174}
CHU_REGALIAS_AJUST	{'n_estimators': 500, 'learning_rate': 0.09909913615269426, 'max_depth': 7, 'min_child_weight': 2, 'subsample': 0.708172616764662, 'colsample_bytree': 0.9508794978508508, 'gamma': 0.49887376061225946, 'reg_alpha': 0.26891921056341245, 'reg_lambda': 0.5637625884315804}

LIGHTGBM

df	best_params
CHU_COPA_AJUST	{'n_estimators': 1670, 'learning_rate': 0.05434009138012196, 'max_depth': 9, 'num_leaves': 43, 'min_child_samples': 27, 'subsample': 0.9577724287559277, 'colsample_bytree': 0.8076995248364037, 'reg_alpha': 0.014464297652987318, 'reg_lambda': 0.6299036642292266}
CHU_REC_PROPIOS_AJUST	{'n_estimators': 1236, 'learning_rate': 0.048922724434304524, 'max_depth': 8, 'num_leaves': 79, 'min_child_samples': 33, 'subsample': 0.7343992892058865, 'colsample_bytree': 0.5125884353625726, 'reg_alpha': 0.09322166284457188, 'reg_lambda': 0.7334602723962557}
CHU_REGALIAS_AJUST	{'n_estimators': 1330, 'learning_rate': 0.09969430566201748, 'max_depth': 9, 'num_leaves': 42, 'min_child_samples': 10, 'subsample': 0.5086757934277606, 'colsample_bytree': 0.8651927912102433, 'reg_alpha': 0.40673825228466354, 'reg_lambda': 0.012809939668950783}