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
# 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]:
```python
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]
titulos = ["CHU_COPA_AJUST", "CHU_REC_PROPIOS_AJUST", "CHU_REGALIAS_AJUST"]
```

In [3]:
```python
# TRAIN TEST
train_copa = dataframes[0].iloc[:round(len(dataframes[0])*.8)]
test_copa = dataframes[0].iloc[round(len(dataframes[0])*.8):]
print(f"Coparticipacion: train({train_copa.shape}), test({test_copa.shape})")

train_recursos = dataframes[1].iloc[:round(len(dataframes[1])*.8)]
test_recursos = dataframes[1].iloc[round(len(dataframes[1])*.8):]
print(f"Recursos: train({train_recursos.shape}), test({test_recursos.shape})")

train_regalias = dataframes[2].iloc[:round(len(dataframes[2])*.8)]
test_regalias = dataframes[2].iloc[round(len(dataframes[2])*.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 [ ]:
```python
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
```

```python
    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_param
    print(f"Entrenamiento de {dataframes_train[i].name} finalizado")
```

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

Out[12]:

	df	best_params
0	CHU_COPA_AJUST	{'changepoint_prior_scale': 0.1, 'seasonality_...
1	CHU_REC_PROPIOS_AJUST	{'changepoint_prior_scale': 0.001, 'seasonalit...
2	CHU_REGALIAS_AJUST	{'changepoint_prior_scale': 0.5, 'seasonality_...

In [13]:
```python
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.
  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=fourie
        model.fit(df)

        df_cv = cross_validation(model, initial='730 days', period='180 days', h
        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
        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': 2114568.5994672766}, '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': 5786977.886510033}}
```

Out[35]:

	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}

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.
  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 [ ]:

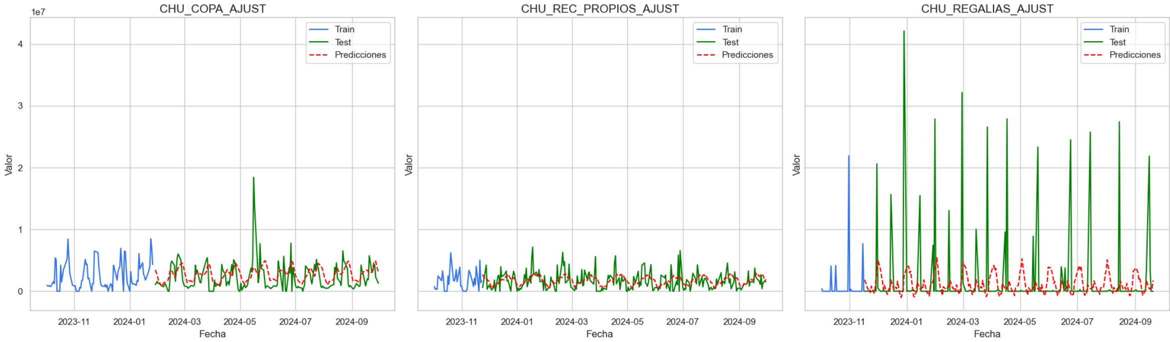
```
results_train_test = []
predictions_test = []
from prophet.make_holidays import make_holidays_df
best_params = pd.read_csv("prophet_best_params.csv")
best_fourier = pd.read_csv("best_fourier_orders.csv")

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(d
    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



CHU_COPA_AJUST
CHU_REC_PROPIOS_AJUST
CHU_REGALIAS_AJUST
1e7
Train
Test
Predicciones
Valor
Fecha
2023-11
2024-01
2024-03
2024-05
2024-07
2024-09

None

In [ ]:

XGBOOST

TRAIN 90%

TEST 10%

In [4]:
```python
# 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]:
```python
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.
  dataframes[i] = dataframes[i].fillna(0)

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

In [7]:
```python
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['
    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]:
```python
# 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})")
```

```python
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))
```

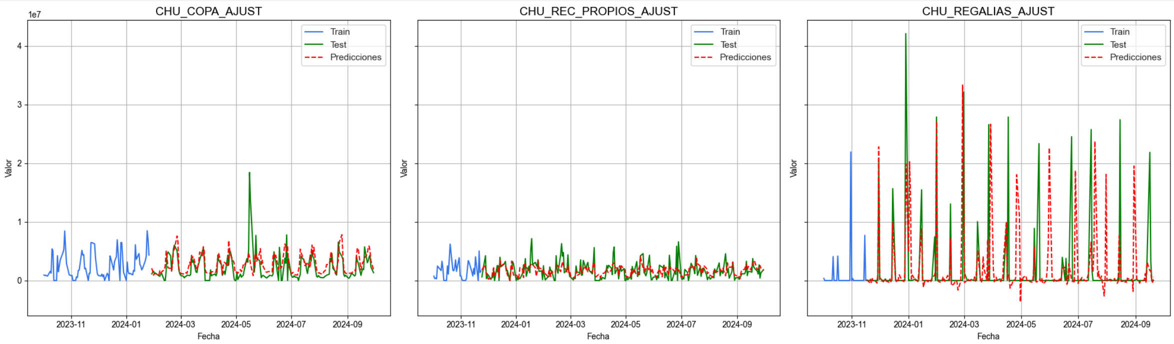
In [ ]:

```python
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(d
    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	696,802,9(
1	XGBRegressor(base_score=None, booster=None, ca...	CHU_REC_PROPIOS_AJUST	1995	222	390,317,6(
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.
  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['
    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)]
```

```python
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 ]
```

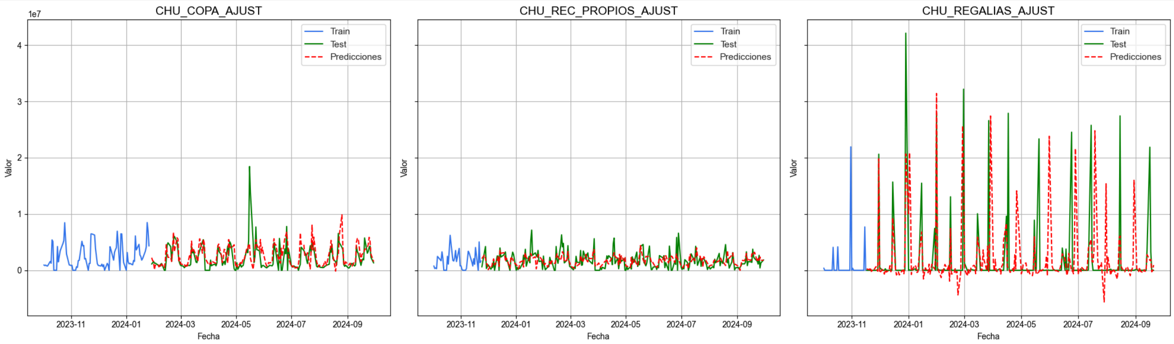
```
Coparticipacion: train((1584, 11)), test((176, 11))
Recursos: train((1995, 11)), test((222, 11))
Regalias: train((1985, 11)), test((221, 11))
```

In [ ]:

```python
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(d
    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_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.
  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,))
```

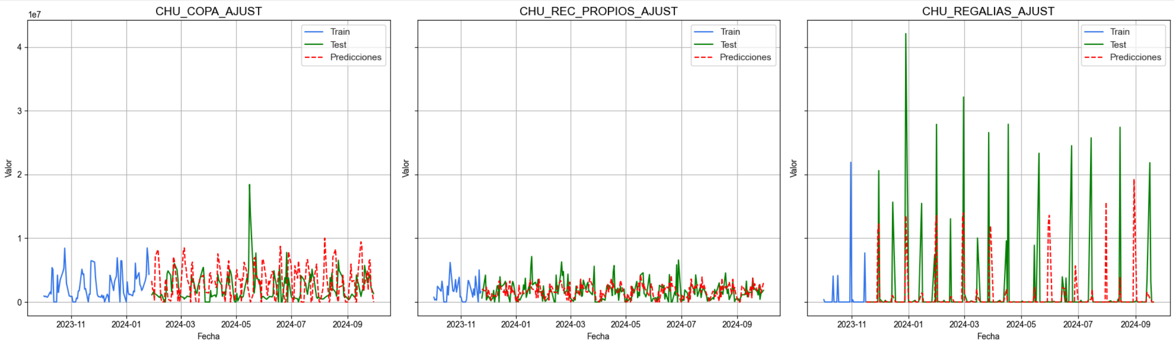
In [ ]:
```
results_train_test = []
predictions_test = []
for i, df in enumerate(dataframes_train):
    df_train = df
    df_test = dataframes_test[i]

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

```python
        max_generations=10,   # intenta optimizar el modelo a traves de 10 itera
        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(), f
    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(d
    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]:

```python
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': 'MaxAbsSc
aler'}, '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 [ ]: