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
```

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

In [ ]:
```

```