

# Informe de los modelos de aplicados en la cuenca: Brujo - Guardia

Jose Andrey Prado Rojas

## Contenidos

<b>1</b>	<b>Base de datos</b>	<b>1</b>
<b>2</b>	<b>Análisis de Datos</b>	<b>2</b>
2.0.1	Brujo . . . . .	3
2.0.2	Guardia . . . . .	6
<b>3</b>	<b>Modelos</b>	<b>10</b>
3.1	Modelo Random Forest . . . . .	11
3.1.1	Brujo . . . . .	11
3.1.2	Guardia . . . . .	15
3.2	Modelo Redes Neuronales . . . . .	19
3.2.1	Brujo . . . . .	19
3.2.2	Guardia . . . . .	22
3.3	Modelo XGBoost . . . . .	24
3.3.1	Brujo . . . . .	24
3.3.2	Guardia . . . . .	41
<b>4</b>	<b>Notas</b>	<b>57</b>
<b>5</b>	<b>Anexos</b>	<b>57</b>

## 1 Base de datos

Entre las variables seleccionadas para el análisis predictivo del caudal mínimo tanto de Brujo como de Guardia se usaron varibales como caudal mínimo, temperatura, precipitación, humedad, humedad del suelo, entre otras. Las variables fueron descargadas usando las

coordenadas exactas de la cuenca según los boletines del ICE y en la base satelital NASA Power se descargaron las siguientes variables

- Humedad relativa a 2 metros de altura en (%)
- Velocidad del viento a 2 metros de altura en (m/s)
- Dirección de la velocidad del viento a 2 metros de altura en grados ( $^{\circ}$ )
- Suma de la precipitación mensual en (mm/día)
- Humedad del perfil del suelo
- Humedad superficial del suelo
- Temperatura a 2 metros de altura en grados Celsius
- Temperatura máxima a 2 metros de altura en grados Celsius
- Temperatura mínima a 2 metros de altura en grados Celsius

Para ambas cuencas se buscó cual era la primera fila completamente llena, esto debido a que los rezagos para las variables fuera de caudal mínimo e índice de niño iban a presentar una serie de valores nulos, también se buscó cual era la última fila cuyos valores fueran llenos, esto debido a que los datos del caudal mínimo terminan en aproximadamente 1993 mientras que las variables descargadas por parte de NASA Power llegan hasta 2025

*Nota:* En las siguientes secciones solo se presentarán los resultados de los módulos y scripts programados con una pequeña explicación ya sea del resultado o bien de la ubicación del código en las carpetas.

```
# Chunk necesario para la importación del código de Python en qmd
library(reticulate)
use_python("C:/Users/andre/anaconda3/envs/Proyectos/python.exe", required = TRUE)
```

## 2 Análisis de Datos

En esta sección se graficarán las series de tiempo del caudal mínimo para ambas cuencas, así como un análisis de estacionalidad y correlaciones.

```
import sys
import os
import importlib

sys.path.append(os.path.abspath('cod/py'))

import AnalisisDatos
import Grafico

importlib.reload(AnalisisDatos)
```

```
importlib.reload(Grafico)
```

```
from AnalisisDatos import AnalisisDatos
from Grafico import Grafico
```

Para el análisis de estadísticas básicas y los outliers se usan los métodos de (AnalisisDatos) y para la visualización de la serie de tiempo, estacionalidad y la correlación se utilizarán los métodos de (Grafico)

### 2.0.1 Brujo

```
datos_brujo = AnalisisDatos('data/Brujo/BaseCompletaBrujo.csv')
grafico_brujo = Grafico('data/Brujo/BaseCompletaBrujo.csv')
```

```
import pandas as pd
resumen = datos_brujo.est_basicas()
resumen = pd.DataFrame(resumen)
resumen = resumen.style.set_caption("Resumen con estadísticas básicas").format(precision=2).background
resumen
```

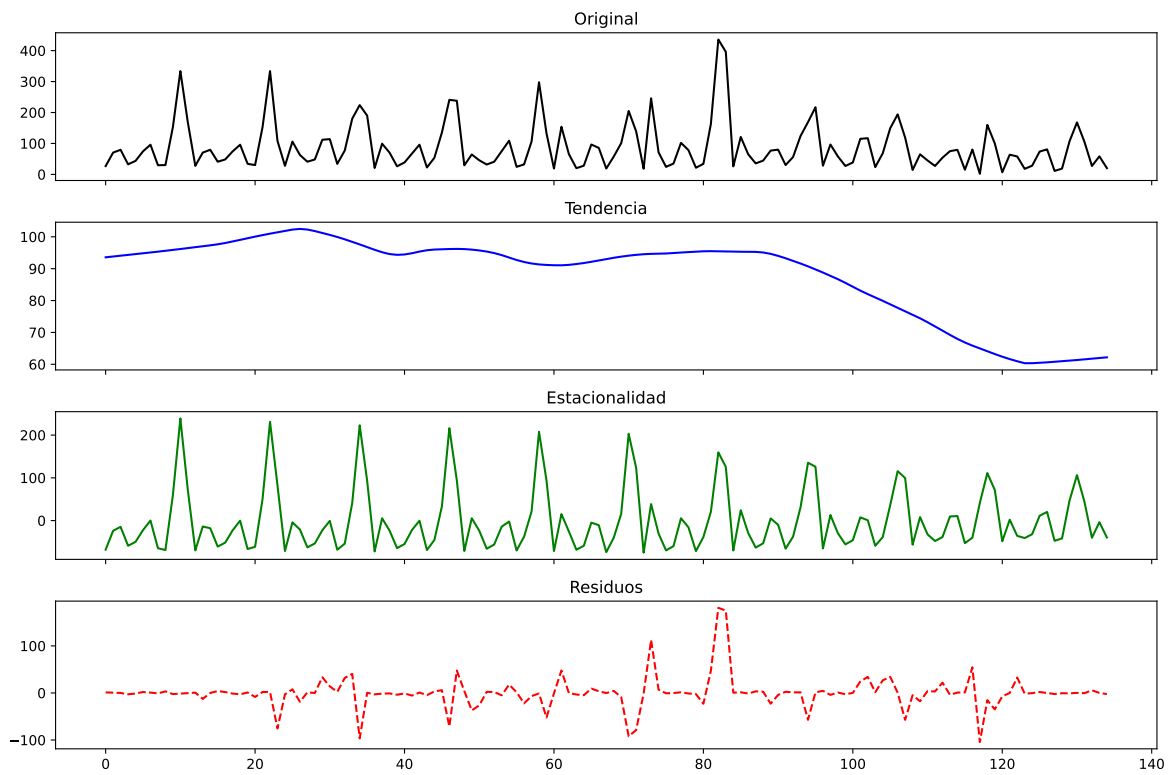
	caudal_minimo	humedad_lag_1	temp_lag_1	temp_max_lag_1	temp_min_lag_1	prep_lag_1
min	2.00	11.38	20.42	24.41	13.80	0.78
q1	31.05	13.68	21.05	26.61	16.40	40.74
q2	67.90	14.89	21.48	27.36	17.04	97.85
q3	108.50	15.30	22.48	29.98	17.69	172.50
max	436.00	16.04	24.05	33.07	19.25	374.92

```
outliers = datos_brujo.detectar_outliers()
outliers = pd.DataFrame(outliers)
outliers = outliers.style.set_caption("Cantidad de Outliers").format(precision=2).background
outliers
```

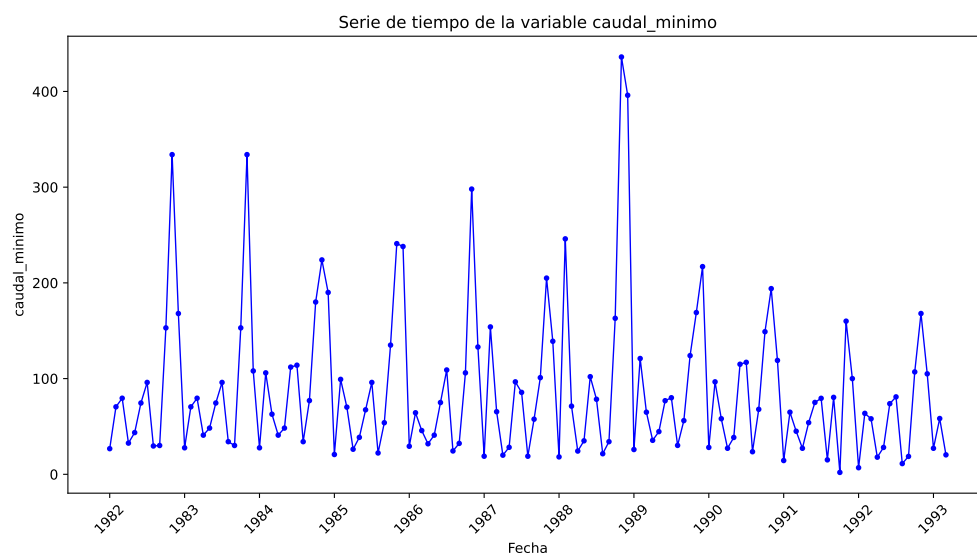
	caudal_minimo	temp_min_lag_1	prep_lag_1	dir_viento_lag_1
Índices	[10, 22, 46, 47, 58, 73, 82, 83]	[29, 52, 68, 77]	[59]	[3, 5, 8, 17, 27, 40, 41, 44, 53, 56]
Cantidad	8	4	1	20

```
grafico_brujo.descomposicion('STL', 'caudal_minimo')
```

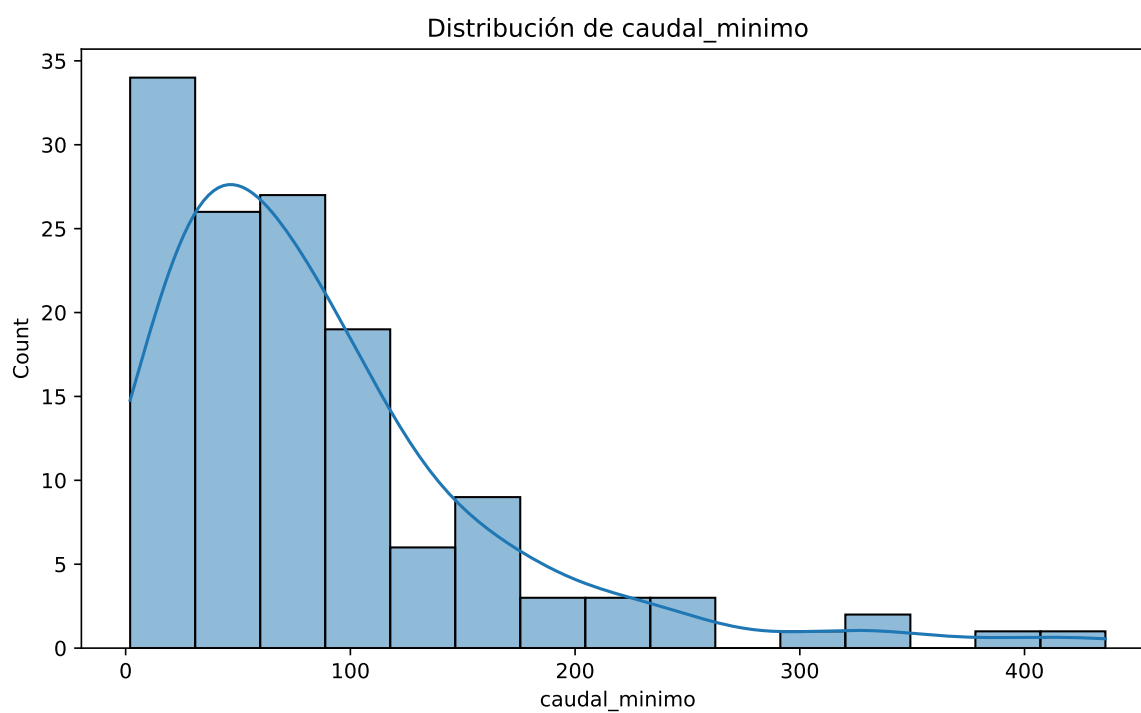
<statsmodels.tsa.seasonal.DecomposeResult object at 0x0000029F2FD88090>



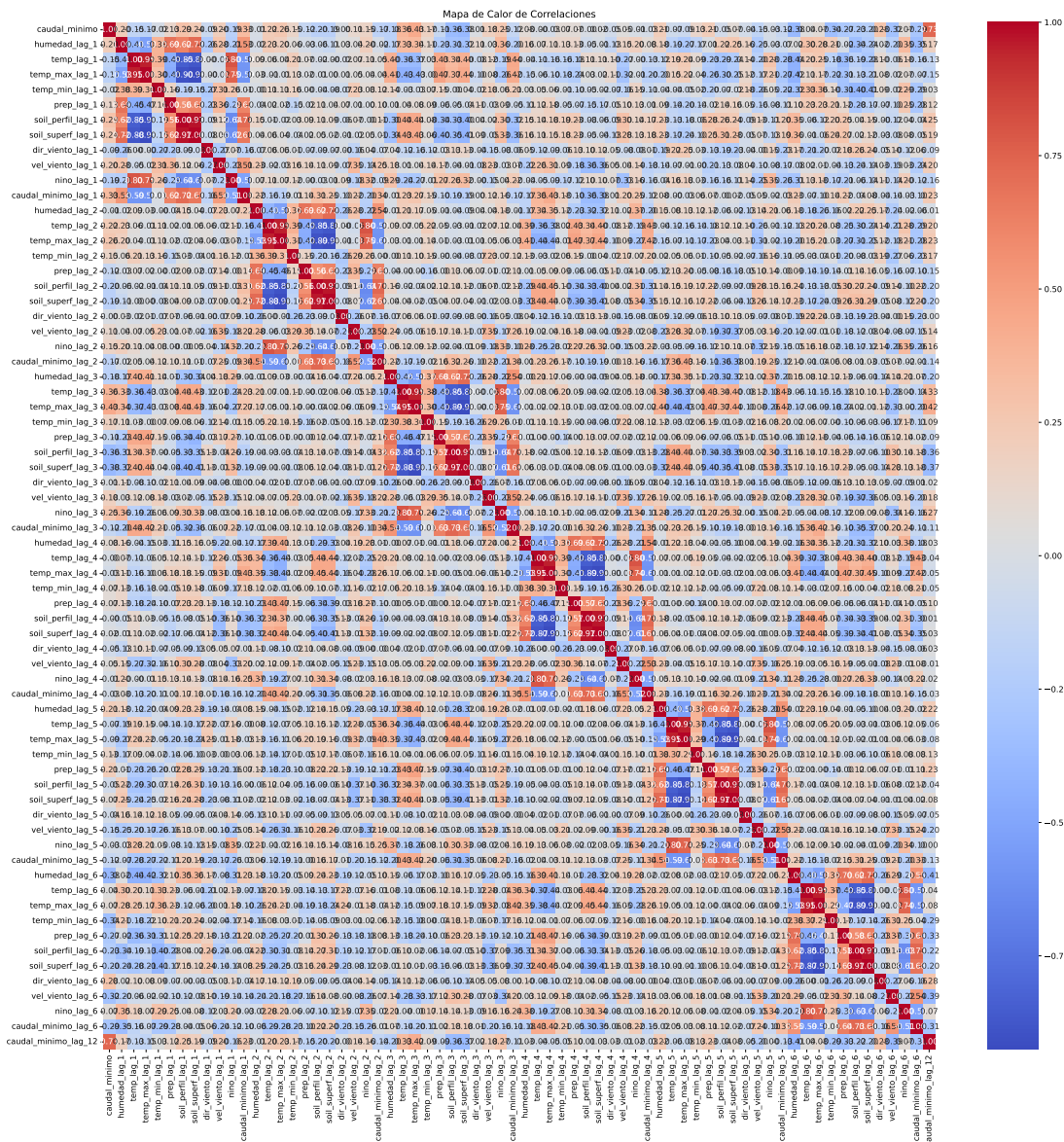
```
grafico_brujo.linea('fecha', 'caudal_minimo')
```



```
grafico_brujo.dist('caudal_minimo')
```



## grafico\_brujo.heatmap()



## 2.0.2 Guardia

```
datos_guardia = AnalisisDatos('data/Guardia/BaseCompletaGuardia.csv')  
grafico_guardia = Grafico('data/Guardia/BaseCompletaGuardia.csv')
```

```
import pandas as pd
resumen = datos_guardia.est_basicas()
resumen = pd.DataFrame(resumen)
resumen = resumen.style.set_caption("Resumen con estadísticas básicas").format(precision=2).background_gradient(cmap=cm.viridis)
resumen
```

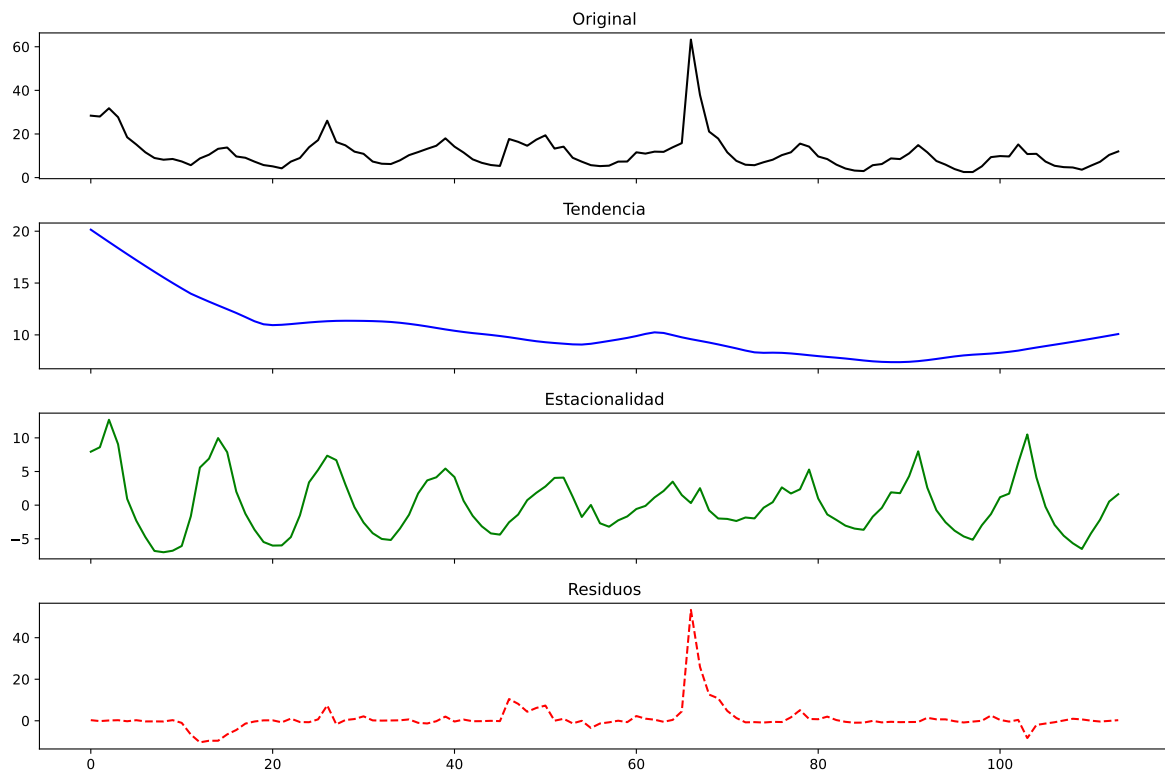
	caudal_minimo	humedad_lag_1	humedad_lag_2	humedad_lag_3	humedad_lag_4	humedad_lag_5
min	2.55	55.35	55.35	55.35	55.35	55.35
q1	6.46	62.24	62.44	62.24	62.24	61.67
q2	9.69	69.83	69.83	69.55	69.13	69.13
q3	13.90	75.56	75.56	75.49	75.49	75.49
max	63.30	86.80	86.80	86.80	86.80	86.80

```
outliers = datos_guardia.detectar_outliers()
outliers = pd.DataFrame(outliers)
outliers = outliers.style.set_caption("Cantidad de Outliers").format(precision=2).background_gradient(cmap=cm.viridis)
outliers
```

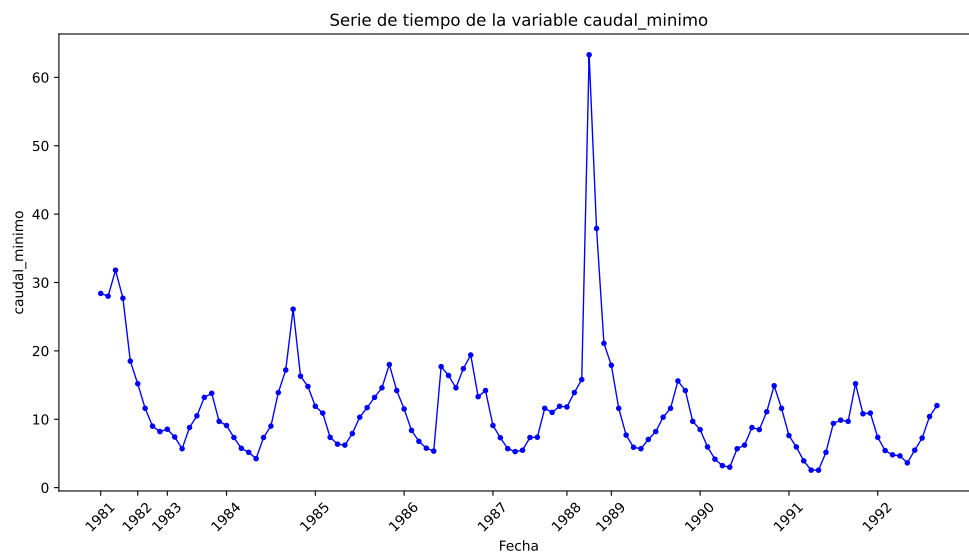
	caudal_minimo	temp_lag_1	temp_lag_3	temp_lag_6	temp_min_lag_1	temp_min_lag_3
Índices	[0, 1, 2, 3, 26, 66, 67]	[11]	[12]	[14]	[18]	[19]
Cantidad	7	1	1	1	1	1

```
grafico_guardia.descomposicion('STL', 'caudal_minimo')
```

```
<statsmodels.tsa.seasonal.DecomposeResult object at 0x0000029F32E507D0>
```

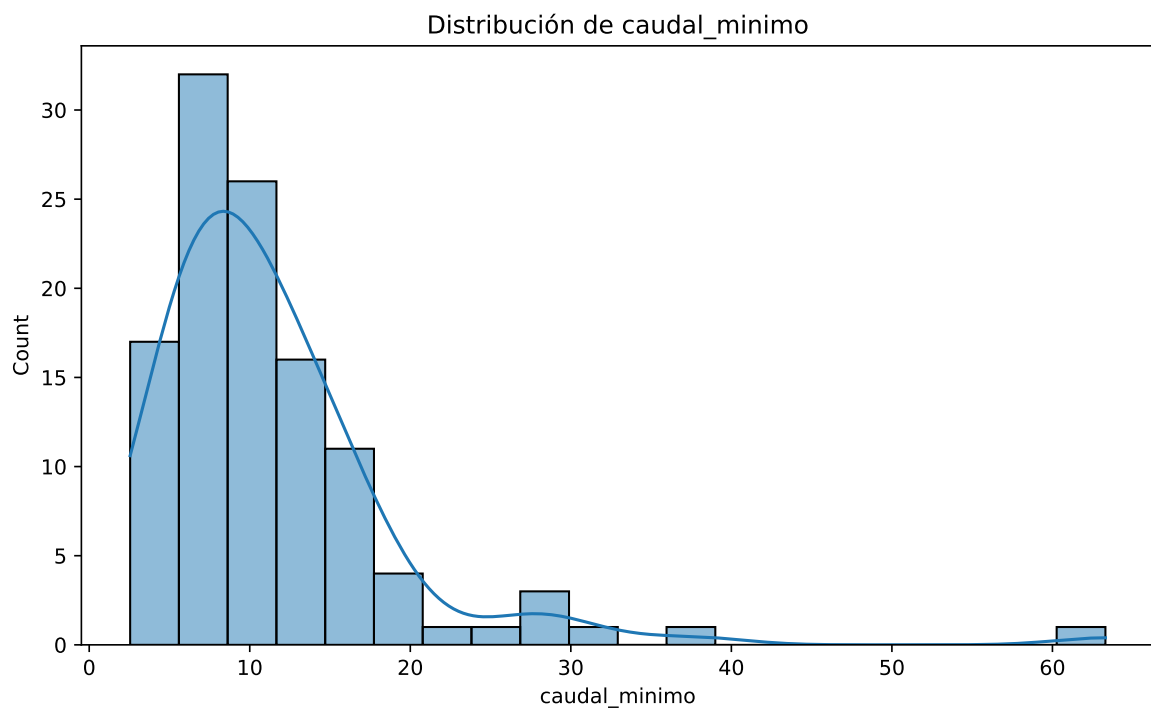


```
grafico_guardia.linea('fecha','caudal_minimo')
```

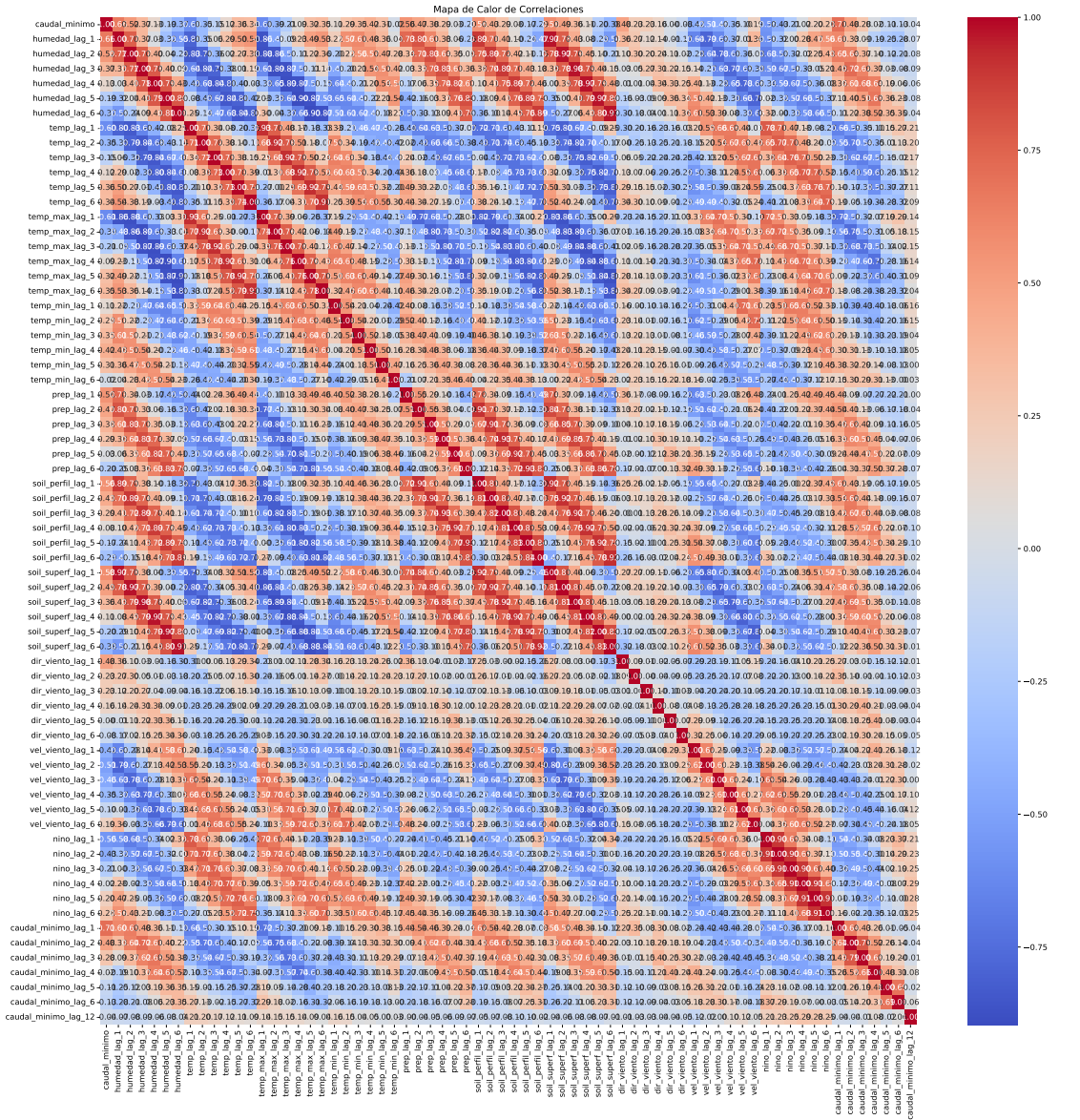




```
grafico_guardia.dist('caudal_minimo')
```



```
grafico_guardia.heatmap()
```



### 3 Modelos

Para el análisis predictivo del caudal mínimo para ambas cuencas se programaron 3 mdoelos, Random Forest, Redes Neuronales y XGBoost, a todos ellos se les aplicó un método de Validación Cruzada y para el caso de XGBoost y RandomForest se aplicó la optimización de los hiperparámetros, en el caso de Redes Neuronales no se aplicaron por su duración a la hora de compilar.

```
import ModeloRandomForest
import RedesNeuronales

importlib.reload(ModeloRandomForest)
```

```
<module 'ModeloRandomForest' from 'C:\\Users\\andre\\OneDrive\\Documentos\\CIMPA-UCR\\Cuencas\\'
```

```
importlib.reload(RedesNeuronales)
```

```
<module 'RedesNeuronales' from 'C:\\Users\\andre\\OneDrive\\Documentos\\CIMPA-UCR\\Cuencas\\'
```

```
from ModeloRandomForest import ModeloRandomForest
from RedesNeuronales import RedesNeuronales

rf_brujo = ModeloRandomForest('data/Brujo/BaseCompletaBrujo.csv', 'caudal_minimo', 'caudal_m
rf_guardia = ModeloRandomForest('data/Guardia/BaseCompletaGuardia.csv', 'caudal_minimo', 'ca
rn_brujo = RedesNeuronales('data/Brujo/BaseCompletaBrujo.csv')
rn_guardia = RedesNeuronales('data/Guardia/BaseCompletaGuardia.csv')
```

```
source('cod/r/XGBoost.R')
```

### 3.1 Modelo Random Forest

#### 3.1.1 Brujo

```
rf_brujo.fechas_num('fecha')
rf_brujo.crear_y_ajustar_modelo()
```

Fitting 12 folds for each of 24 candidates, totalling 288 fits

Mejores parámetros {'bootstrap': True, 'max\_depth': 10, 'max\_features': 'sqrt', 'min\_samples

Out-of-bag Score (OOB): 0.5781

```
rf_brujo.evaluar_modelo()
```

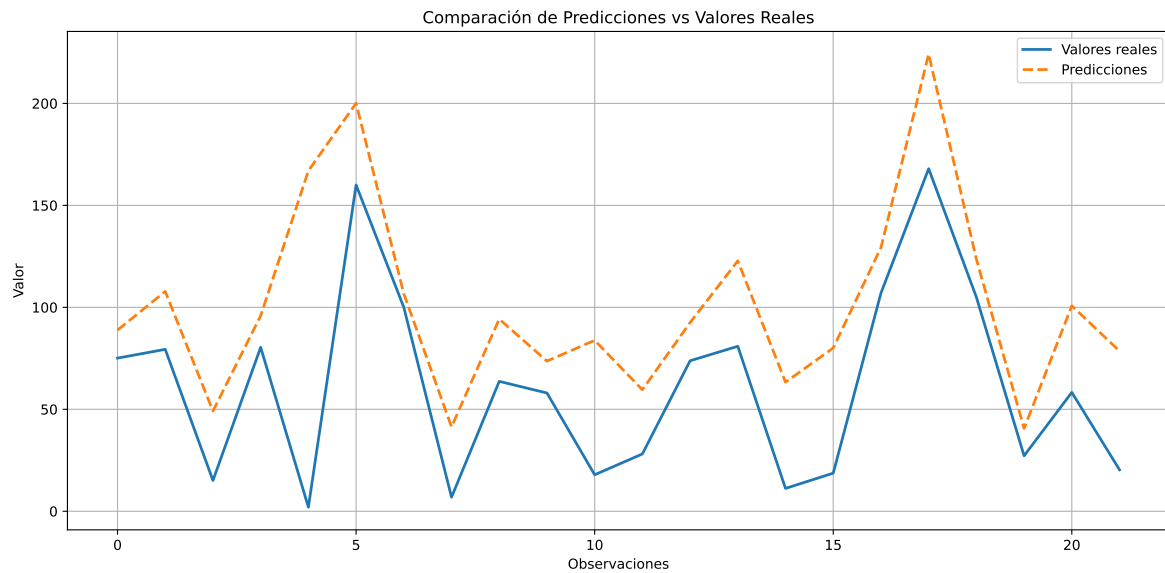
$R^2$  (train): 0.9044

$R^2$  promedio (CV): 0.5370

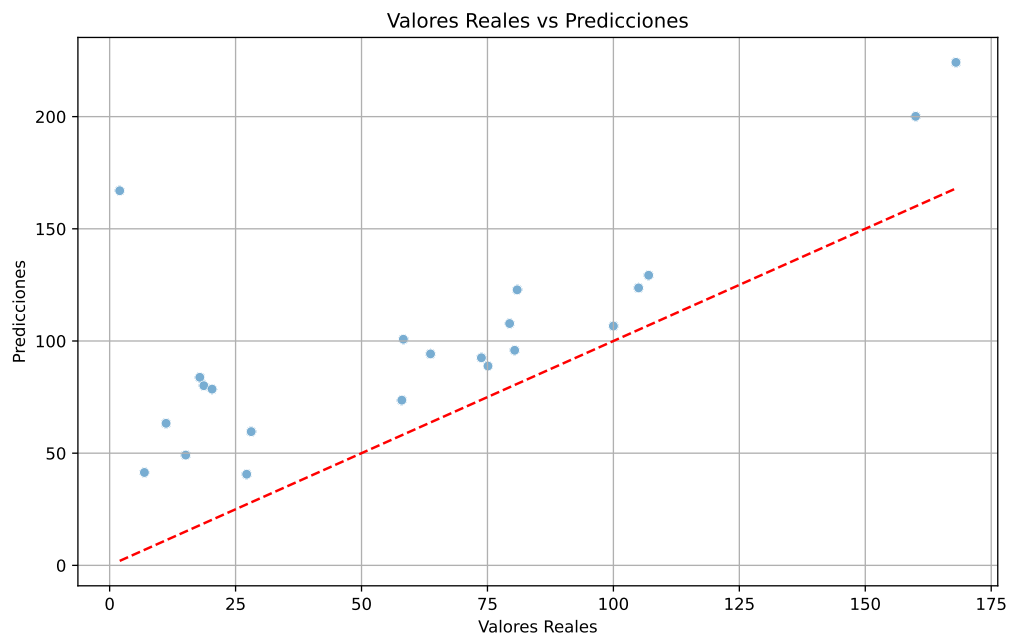
Métricas de Error:

- MAE (Error Absoluto Medio): 39.40
- MSE (Error Cuadrático Medio): 2587.74
- RMSE (Raíz del Error Cuadrático Medio): 50.87
- MAPE (Error Porcentual Absoluto Medio): 499.34%
- NSE (Eficiencia Nash-Sutcliffe): -0.2157,
- MAE Mediana: 32.78

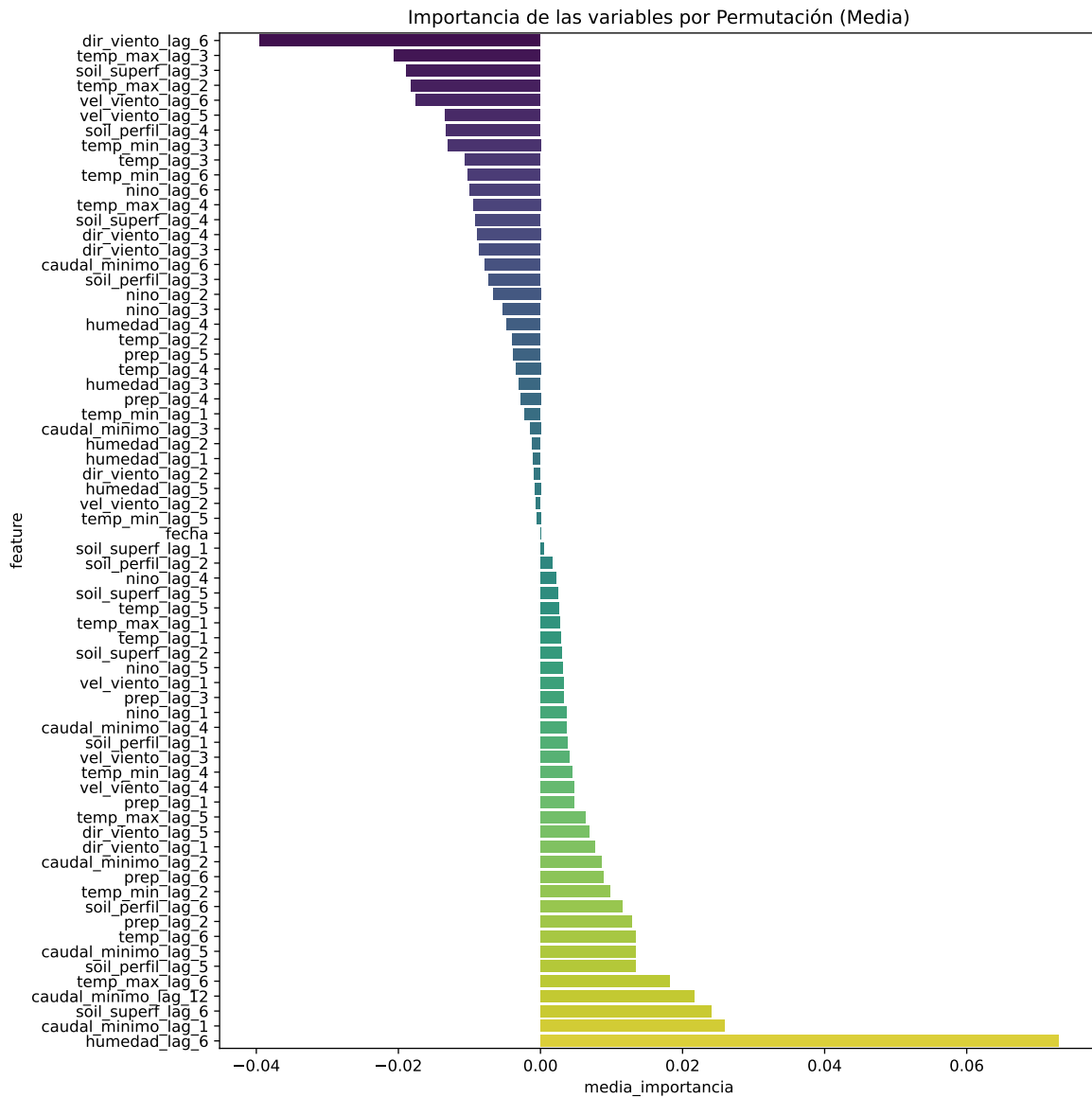
```
rf_brujo.graficar_predicciones()
```



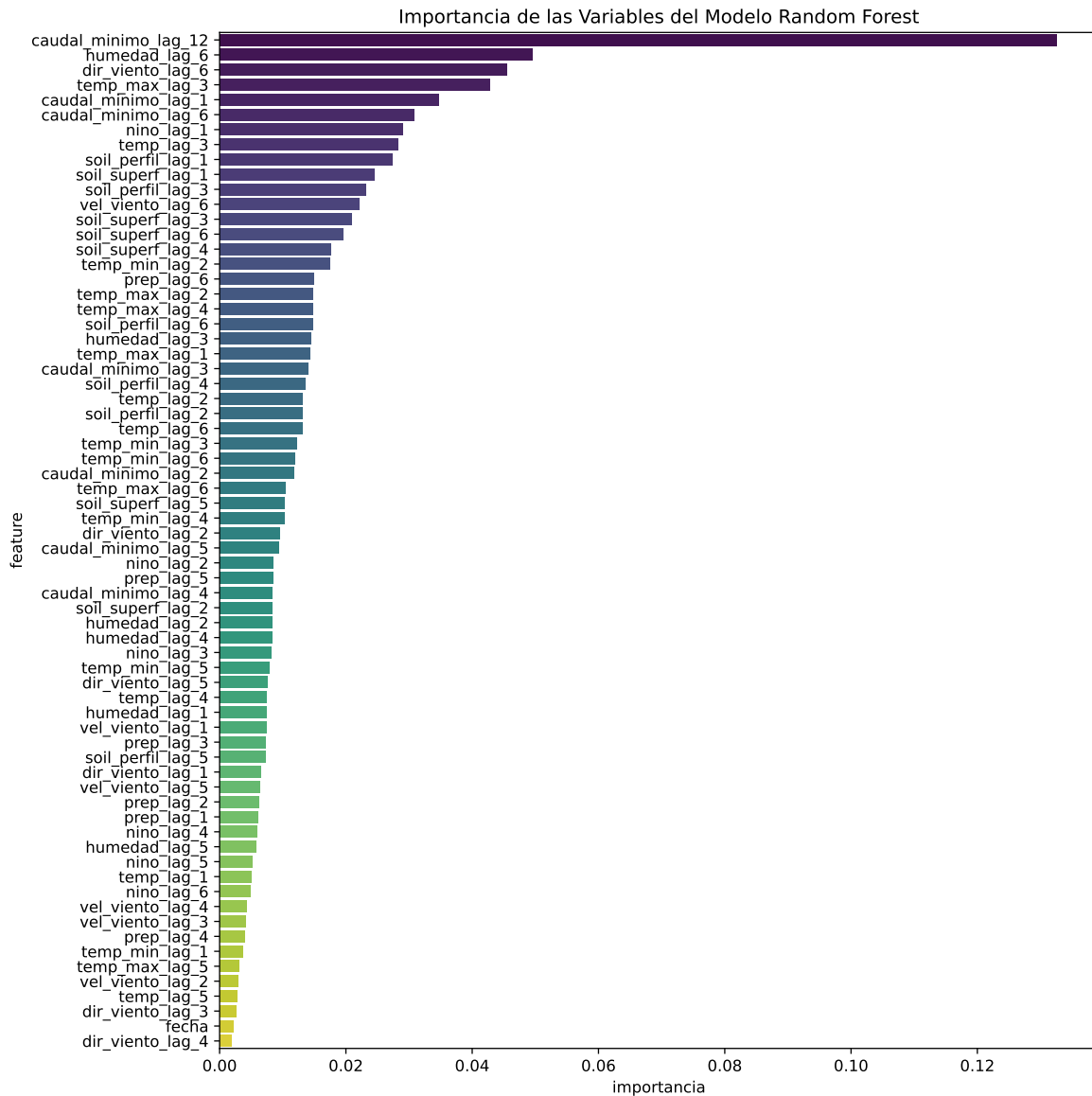
```
rf_brujo.visualizar_resultados()
```



```
rf_brujo.importancia_permutacion('media', 100)
```



```
rf_brujo.importancia_feature(100)
```



### 3.1.2 Guardia

```
rf_guardia.fechas_num('fecha')
rf_guardia.crear_y_ajustar_modelo()
```

Fitting 12 folds for each of 24 candidates, totalling 288 fits

Mejores parámetros {'bootstrap': True, 'max\_depth': 10, 'max\_features': 'sqrt', 'min\_samples': 10, 'min\_samples\_leaf': 10, 'min\_samples\_split': 10, 'n\_estimators': 1000, 'oob\_score': 0.4199, 'random\_state': 42, 'verbose': 0}  
Out-of-bag Score (OOB): 0.4199

```
rf_guardia.evaluar_modelo()
```

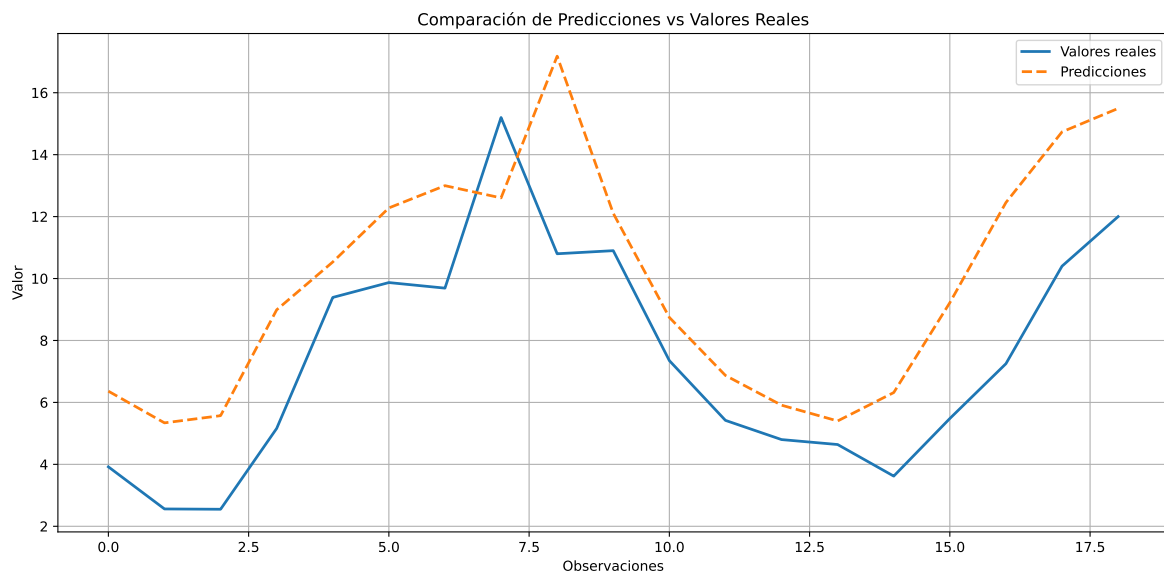
$R^2$  (train): 0.7156

$R^2$  promedio (CV): 0.1145

Métricas de Error:

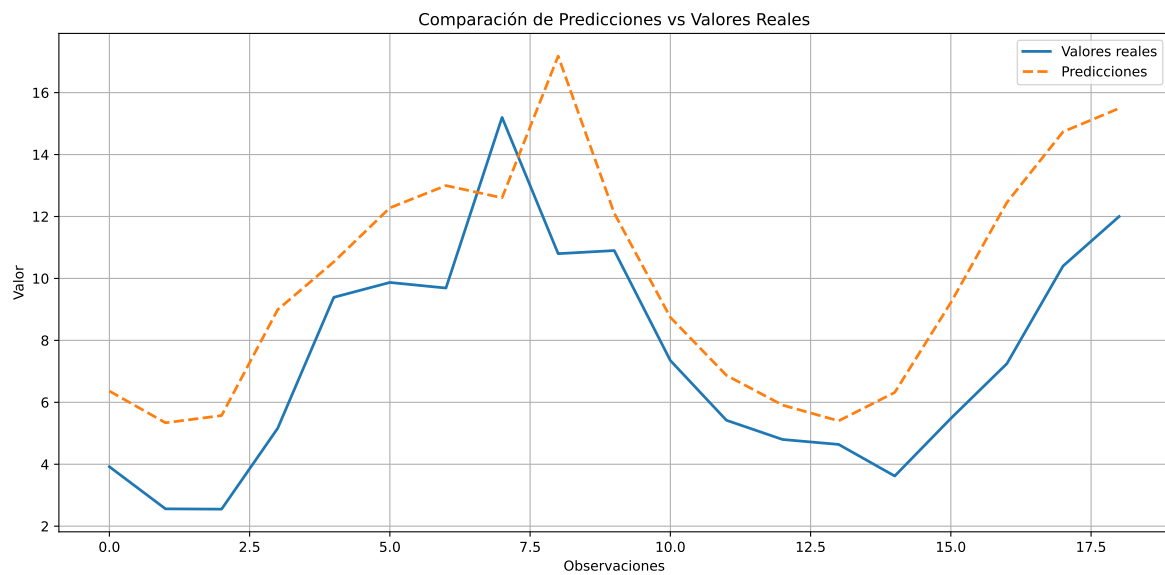
- MAE (Error Absoluto Medio): 2.81
- MSE (Error Cuadrático Medio): 9.98
- RMSE (Raíz del Error Cuadrático Medio): 3.16
- MAPE (Error Porcentual Absoluto Medio): 46.96%
- NSE (Eficiencia Nash-Sutcliffe): 0.1771,
- MAE Mediana: 2.70

```
rf_guardia.graficar_predicciones()
```

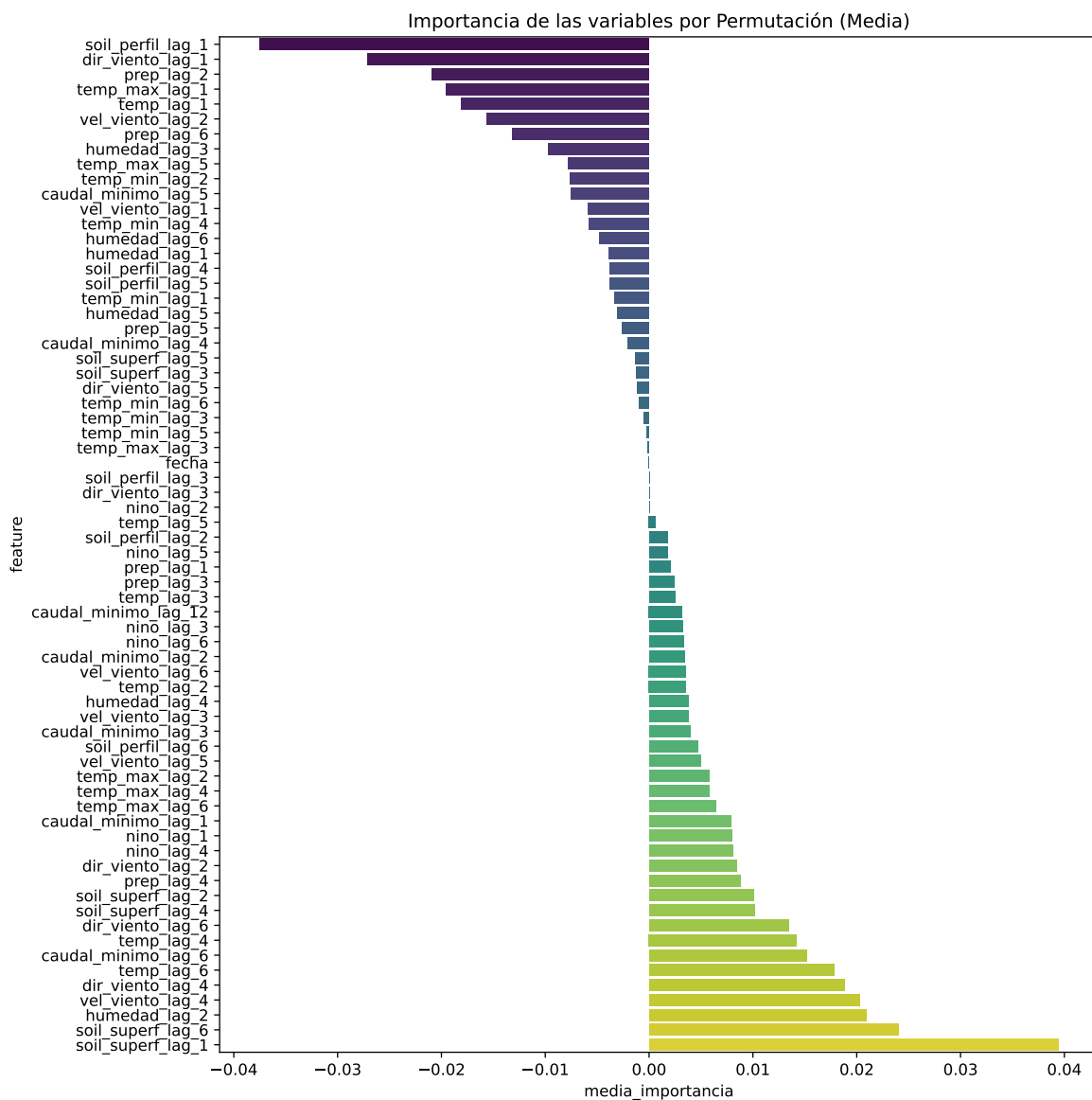


```
rf_guardia.graficar_predicciones()
```

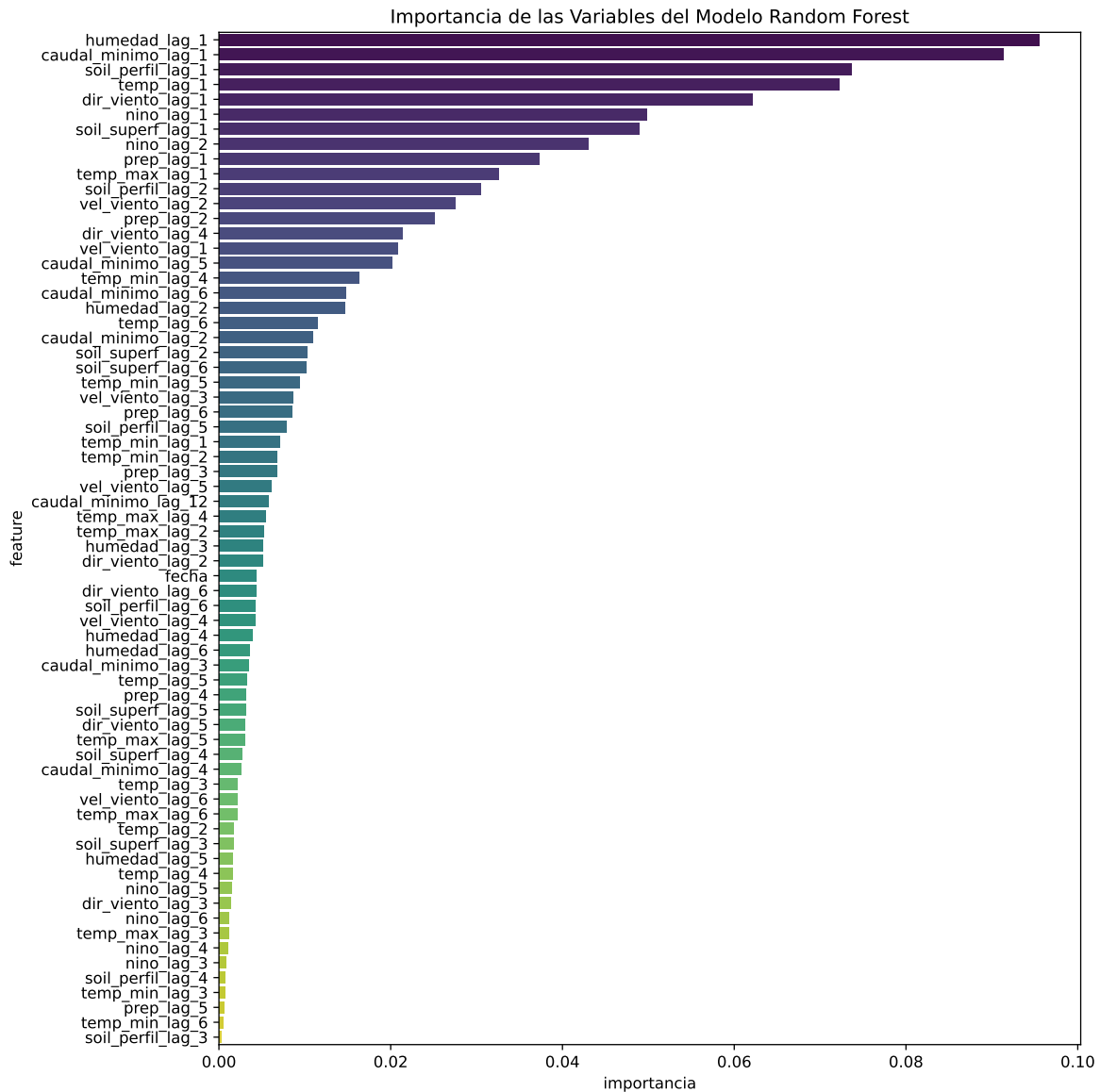




```
rf_guardia.importancia_permutacion('media', 100)
```



```
rf_guardia.importancia_feature(100)
```



## 3.2 Modelo Redes Neuronales

### 3.2.1 Brujo

```
rn_brujo.fechas_num('fecha')
X_train, X_test, y_train, y_test = rn_brujo.preprocesar_datos(['fecha', 'humedad_lag_1', 'humedad_lag_3', 'humedad_lag_4', 'humedad_lag_5', 'humedad_lag_6',
```

```
'temp_lag_1', 'temp_lag_2', 'temp_lag_3', 'temp_lag_4', 'temp_lag_5',
'temp_lag_6', 'temp_max_lag_1', 'temp_max_lag_2', 'temp_max_lag_3',
'temp_max_lag_4', 'temp_max_lag_5', 'temp_max_lag_6', 'temp_min_lag_1',
'temp_min_lag_2', 'temp_min_lag_3', 'temp_min_lag_4', 'temp_min_lag_5',
'temp_min_lag_6', 'prep_lag_1', 'prep_lag_2', 'prep_lag_3',
'prep_lag_4', 'prep_lag_5', 'prep_lag_6', 'soil_perfil_lag_1',
'soil_perfil_lag_2', 'soil_perfil_lag_3', 'soil_perfil_lag_4',
'soil_perfil_lag_5', 'soil_perfil_lag_6', 'soil_superf_lag_1',
'soil_superf_lag_2', 'soil_superf_lag_3', 'soil_superf_lag_4',
'soil_superf_lag_5', 'soil_superf_lag_6', 'dir_viento_lag_1',
'dir_viento_lag_2', 'dir_viento_lag_3', 'dir_viento_lag_4',
'dir_viento_lag_5', 'dir_viento_lag_6', 'vel_viento_lag_1',
'vel_viento_lag_2', 'vel_viento_lag_3', 'vel_viento_lag_4',
'vel_viento_lag_5', 'vel_viento_lag_6', 'nino_lag_1', 'nino_lag_2',
'nino_lag_3', 'nino_lag_4', 'nino_lag_5', 'nino_lag_6',
'caudal_minimo_lag_1', 'caudal_minimo_lag_2', 'caudal_minimo_lag_3',
'caudal_minimo_lag_4', 'caudal_minimo_lag_5', 'caudal_minimo_lag_6',
'caudal_minimo_lag_12'], 'caudal_minimo', 30, 0.2)
```

```
rn_brujo.validacion_cruzada(n_splits=5, epochs=250, batch_size=32)
```

```
rn_brujo.entrenar(epochs=150, batch_size=32, verbose=1, validation_split=0.2)
```

```
rn_brujo.predecir()
```

```
1/1          0s 45ms/step
1/1          0s 56ms/step
array([157.1532 , 25.6714 , 53.21646 , 126.03226 , 269.24966 ,
       157.38629 , 53.897762, 123.59169 , 90.77351 , 60.154274,
       64.71527 , 82.95827 , 132.8184 , 30.040842, 57.324303,
       148.18408 , 232.89346 , 153.82907 , 53.222977, 142.2532 ,
       86.47565 ], dtype=float32)
```

```
rn_brujo.evaluar_modelo()
```

```
R2: -0.5549
```

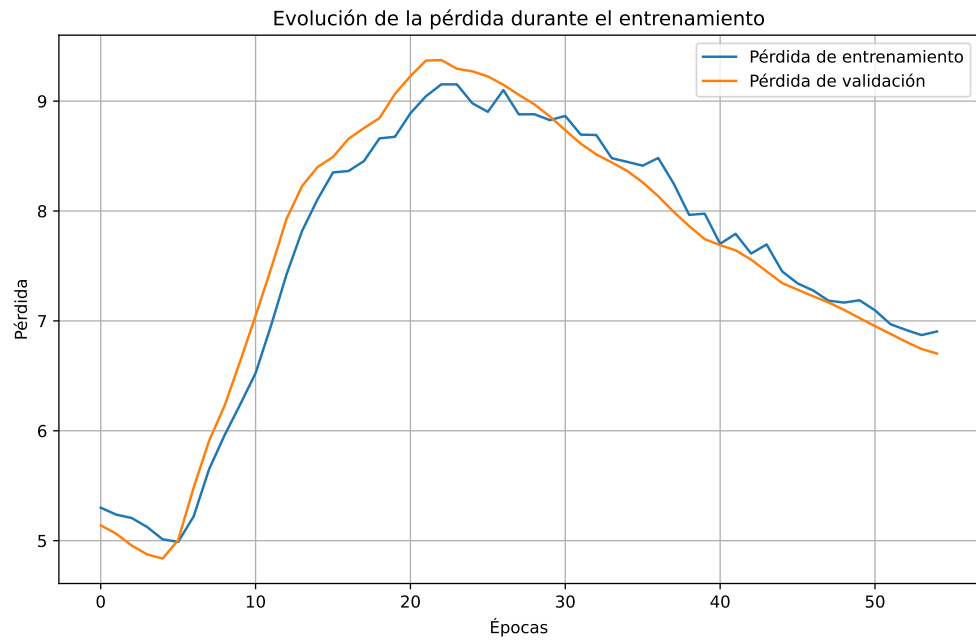
```
RMSE: 58.7661
```

```
MAE: 51.1559
```

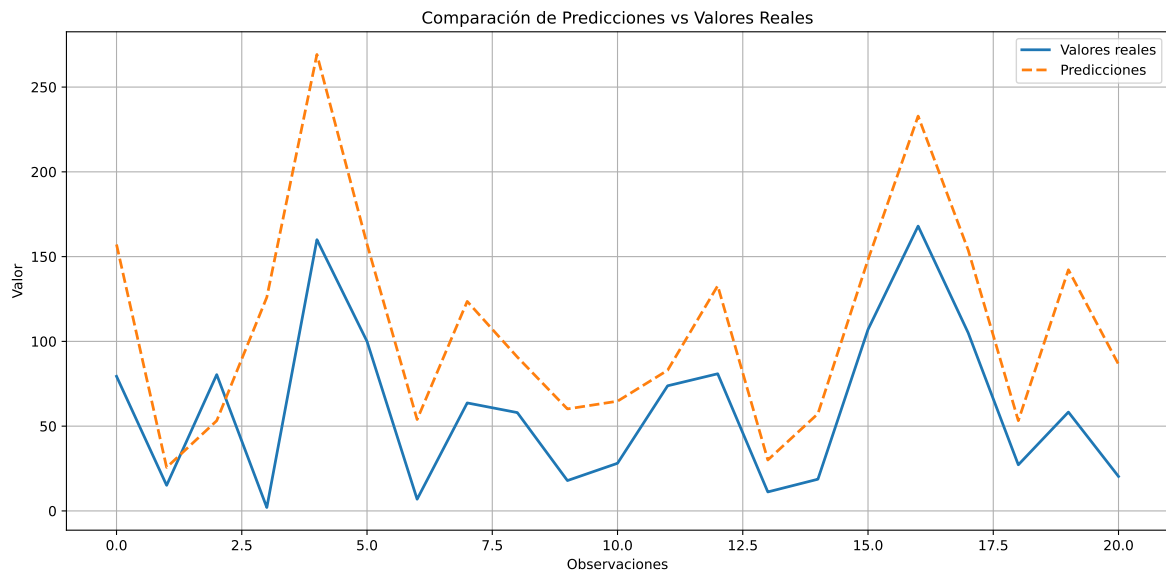
```
NSE: -0.5549
```

```
{'R2': -0.5548948414281789, 'RMSE': np.float64(58.766051289604945), 'MAE': 51.15585075253239
```

```
rn_brujo.graficar_perdidas()
```



```
rn_brujo.graficar_resultados('fecha')
```



### 3.2.2 Guardia

```
rn_guardia.fechas_num('fecha')
X_train, X_test, y_train, y_test = rn_guardia.preprocesar_datos(['fecha', 'humedad_lag_1', 'humedad_lag_3', 'humedad_lag_4', 'humedad_lag_5', 'humedad_lag_6', 'temp_lag_1', 'temp_lag_2', 'temp_lag_3', 'temp_lag_4', 'temp_lag_5', 'temp_lag_6', 'temp_max_lag_1', 'temp_max_lag_2', 'temp_max_lag_3', 'temp_max_lag_4', 'temp_max_lag_5', 'temp_max_lag_6', 'temp_min_lag_1', 'temp_min_lag_2', 'temp_min_lag_3', 'temp_min_lag_4', 'temp_min_lag_5', 'temp_min_lag_6', 'prep_lag_1', 'prep_lag_2', 'prep_lag_3', 'prep_lag_4', 'prep_lag_5', 'prep_lag_6', 'soil_perfil_lag_1', 'soil_perfil_lag_2', 'soil_perfil_lag_3', 'soil_perfil_lag_4', 'soil_perfil_lag_5', 'soil_perfil_lag_6', 'soil_superf_lag_1', 'soil_superf_lag_2', 'soil_superf_lag_3', 'soil_superf_lag_4', 'soil_superf_lag_5', 'soil_superf_lag_6', 'dir_viento_lag_1', 'dir_viento_lag_2', 'dir_viento_lag_3', 'dir_viento_lag_4', 'dir_viento_lag_5', 'dir_viento_lag_6', 'vel_viento_lag_1', 'vel_viento_lag_2', 'vel_viento_lag_3', 'vel_viento_lag_4', 'vel_viento_lag_5', 'vel_viento_lag_6', 'nino_lag_1', 'nino_lag_2', 'nino_lag_3', 'nino_lag_4', 'nino_lag_5', 'nino_lag_6', 'caudal_minimo_lag_1', 'caudal_minimo_lag_2', 'caudal_minimo_lag_3', 'caudal_minimo_lag_4', 'caudal_minimo_lag_5', 'caudal_minimo_lag_6', 'caudal_minimo_lag_12'], 'caudal_minimo', 30, 0.2)

rn_guardia.validacion_cruzada(n_splits=5, epochs=250, batch_size=32)
```

```
rn_guardia.entrenar(epochs=150, batch_size=32, verbose=1, validation_split=0.2)
```

```
rn_guardia.predecir()
```

```
1/1          0s 45ms/step
1/1          0s 56ms/step
array([ 8.143096 ,  8.725153 ,  8.36546  ,  9.786261 , 11.135403 ,
        11.332285 , 10.196749 ,  8.7102585, 13.379669 ,  8.746128 ,
         7.868424 ,  8.436724 ,  8.399451 ,  8.881443 , 10.81162  ,
        10.192707 , 10.68141  ], dtype=float32)
```

```
rn_guardia.evaluar_modelo()
```

R<sup>2</sup>: 0.0162

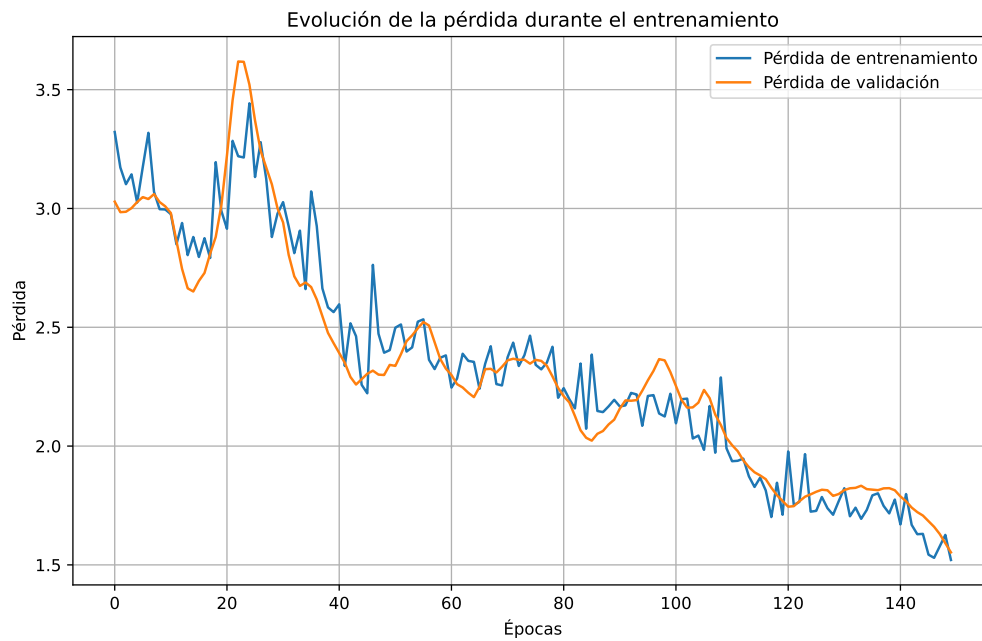
RMSE: 3.3201

MAE: 2.8154

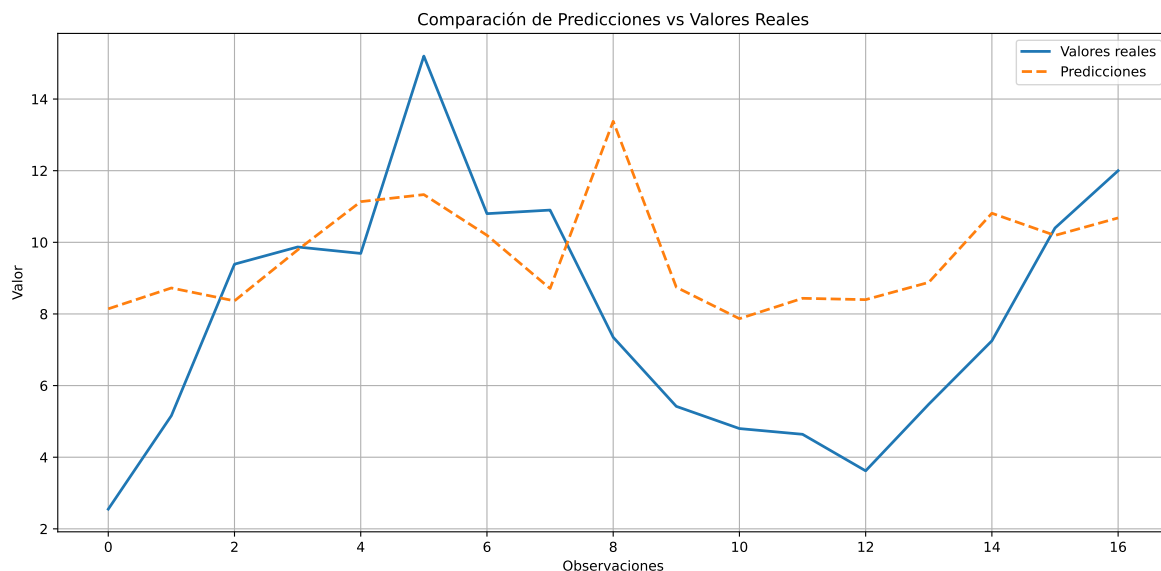
NSE: 0.0162

{'R<sup>2</sup>': 0.01623792615829389, 'RMSE': np.float64(3.320056256460805), 'MAE': 2.8154106196235213

```
rn_guardia.graficar_perdidas()
```



```
rn_guardia.graficar_resultados('fecha')
```



### 3.3 Modelo XGBoost

#### 3.3.1 Brujo

```
xg_brujo <- read.csv('data/Brujo/BaseCompletaBrujo.csv')
xg_brujo$fecha <- as.Date(xg_brujo$fecha)
lista_brujo <- xg_particion_datos(xg_brujo, "caudal_minimo", 0.70)

grid <- expand.grid(
  nrounds = c(100, 300, 500),
  max_depth = c(4, 6, 8),
  eta = c(0.005, 0.01),
  gamma = c(0, 1, 5),
  colsample_bytree = c(0.5, 0.7),
  min_child_weight = c(5, 10),
  subsample = c(0.6, 0.8)
)

modelo_opti_brujo <- xg_optimizar_modelo(lista_brujo, grid, cv = 6)
```

```
[16:44:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:44:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:44:37] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```







[illegible]





[illegible]







[illegible]

[illegible]

[illegible]

[illegible]

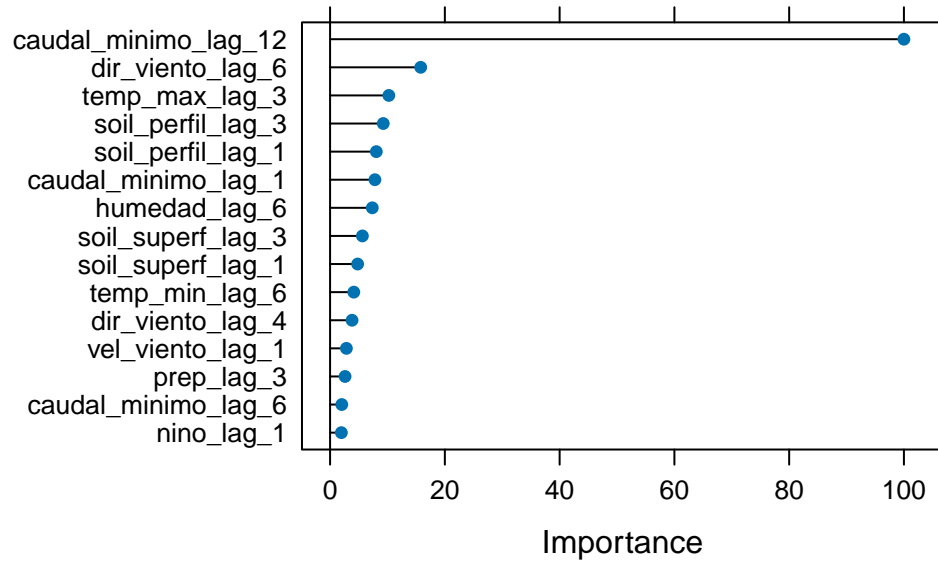


```

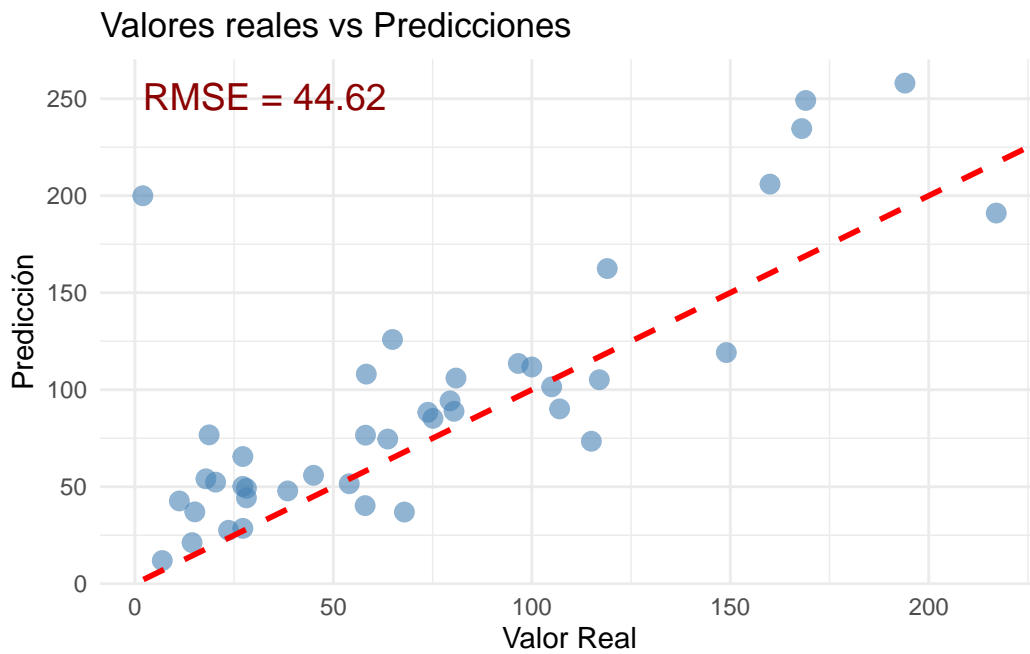
[16:48:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:25] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:26] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:27] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:28] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:29] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
--METRICAS DE ENTRENAMIENTO--
RMSE: 33.12712
MAE: 17.77212
MAPE: 0.2260837
R2: 0.8736209
NSE: 0.7364693
--METRICAS DE PRUEBA--
RMSE: 44.62473
MAE: 30.15685
MAPE 2.989752
R2: 0.6169777
NSE: 0.4950413

```

```
xg_grafico_importancia(modelo_opti_brujo, 15, opti = TRUE)
```

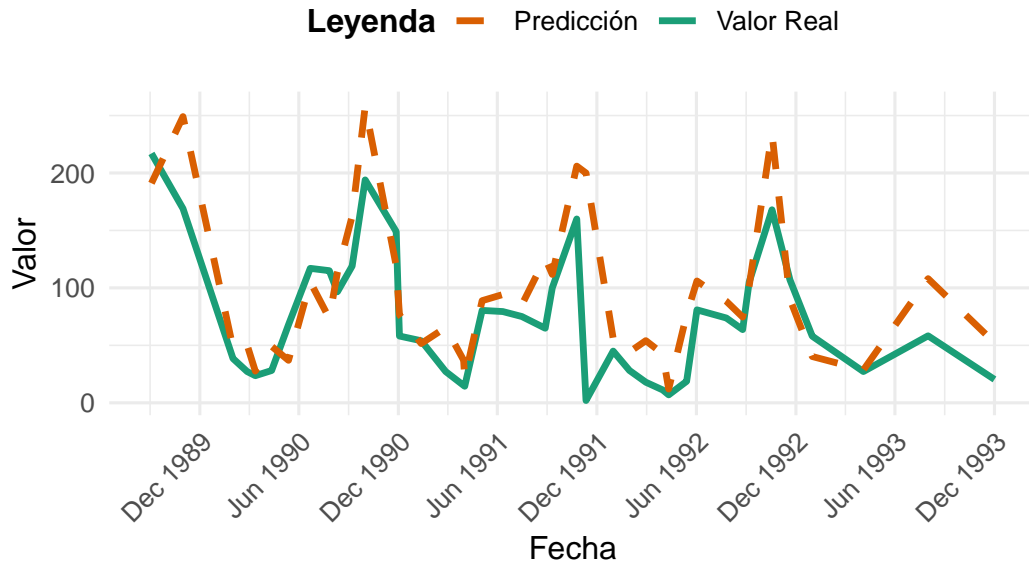


```
xg_grafico_resultados(modelo_opti_brujo, lista_brujo, opti = TRUE)
```

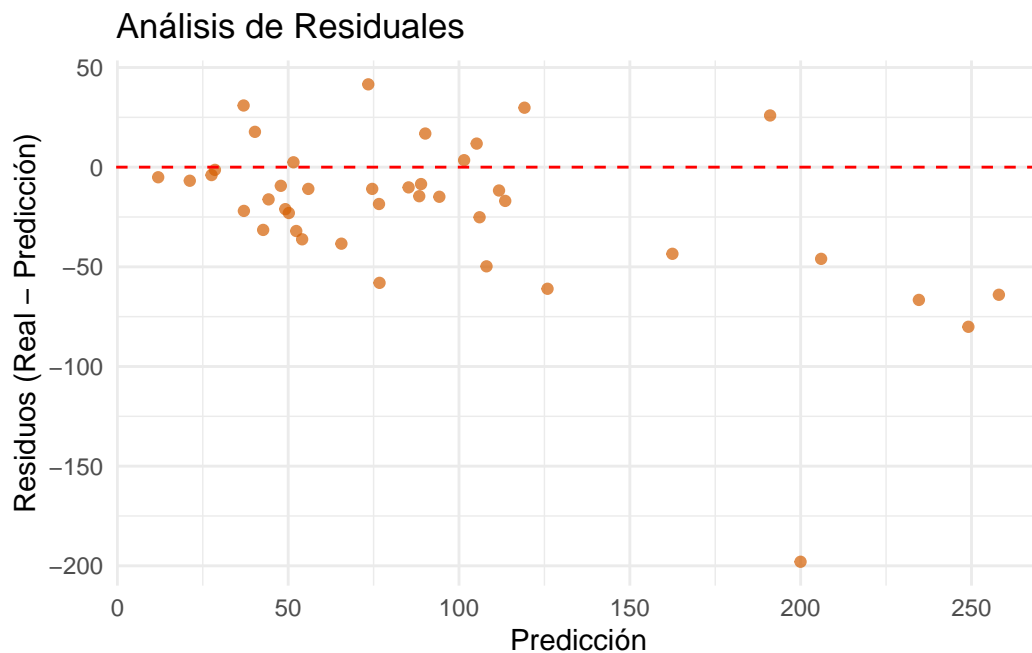


```
xg_serie_tiempo(modelo = modelo_opti_brujo, lista_datos = lista_brujo, nombre_fecha = "fecha")
```

## Comparación entre valores reales y predicciones:



```
xg_grafico_residuos(modelo_opti_brujo, lista_brujo, opti = TRUE)
```





### 3.3.2 Guardia

```
xg_guardia <- read.csv('data/Guardia/BaseCompletaGuardia.csv')
xg_guardia$fecha <- as.Date(xg_guardia$fecha)
lista_guardia <- xg_particion_datos(xg_guardia, "caudal_minimo", 0.70)

grid <- expand.grid(
  nrounds = c(100, 300, 500),
  max_depth = c(4, 6, 8),
  eta = c(0.005, 0.01),
  gamma = c(0, 1, 5),
  colsample_bytree = c(0.5, 0.7),
  min_child_weight = c(5, 10),
  subsample = c(0.6, 0.8)
)

modelo_opti_guardia <- xg_optimizar_modelo(lista_guardia, grid, cv = 6)
```

```
[16:48:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:30] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:31] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:32] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:33] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:33] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:33] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:33] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:33] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:34] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:35] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
[16:48:36] WARNING: src/c_api/c_api.cc:935: `ntree_limit` is deprecated, use `iteration_range`
```

[illegible]





[illegible]

[illegible]

[illegible]







[illegible]



[illegible]





--METRICAS DE PRUEBA--

RMSE: 3.42352

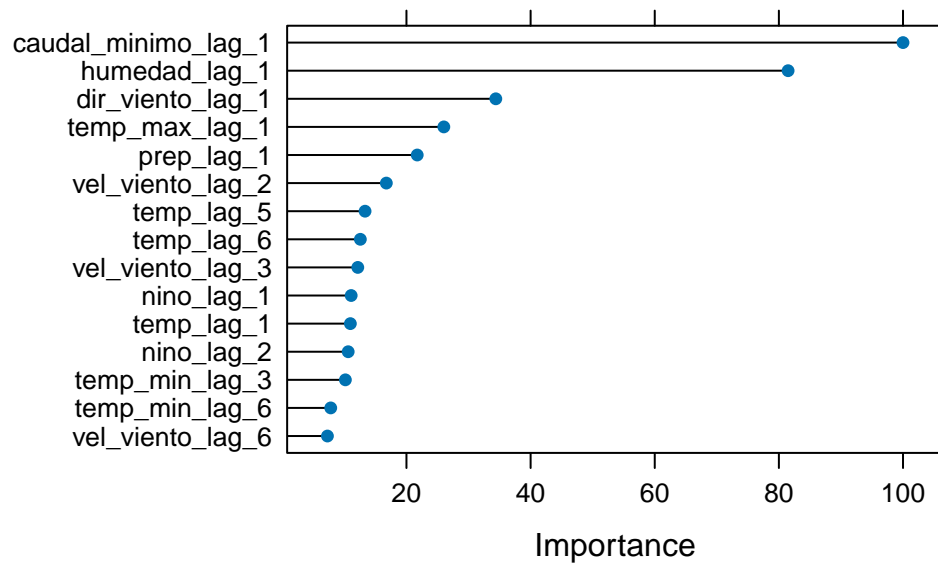
MAE: 2.848145

MAPE 0.4663932

$R^2$ : 0.6622503

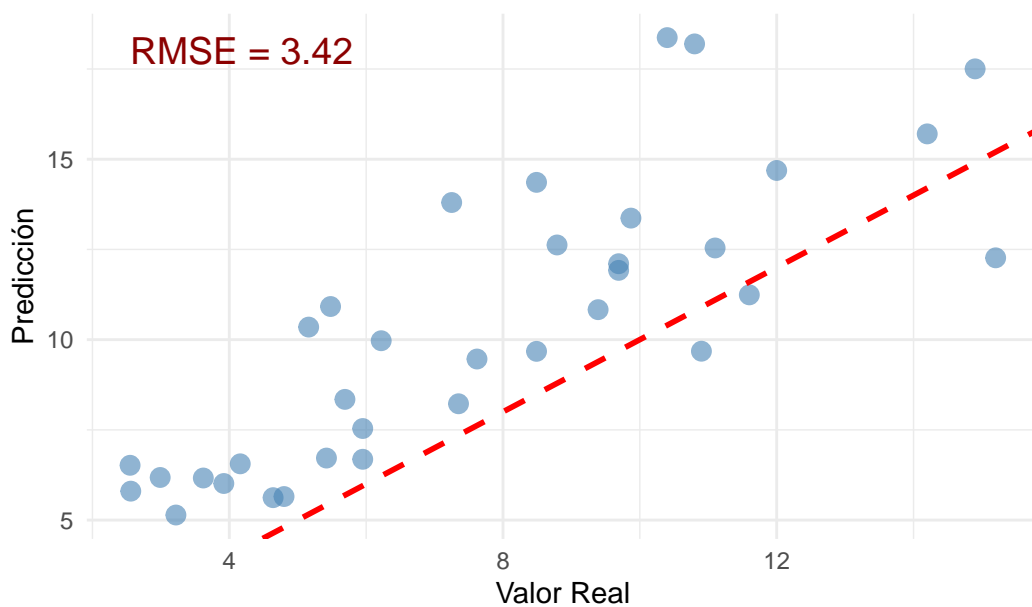
NSE: 0.1828486

```
xg_grafico_importancia(modelo_opti_guardia, 15, opti = TRUE)
```



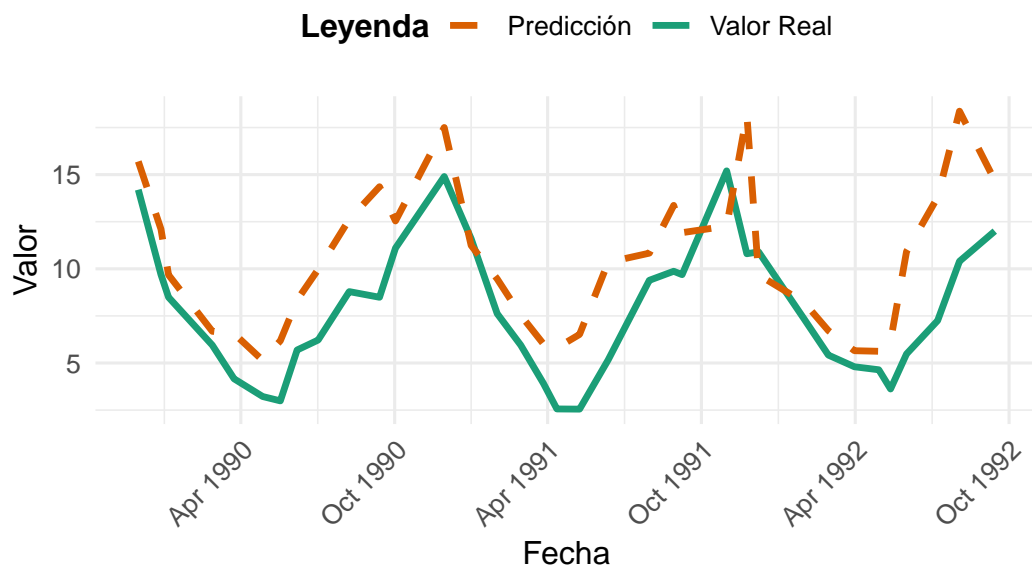
```
xg_grafico_resultados(modelo_opti_guardia, lista_guardia, opti = TRUE)
```

### Valores reales vs Predicciones



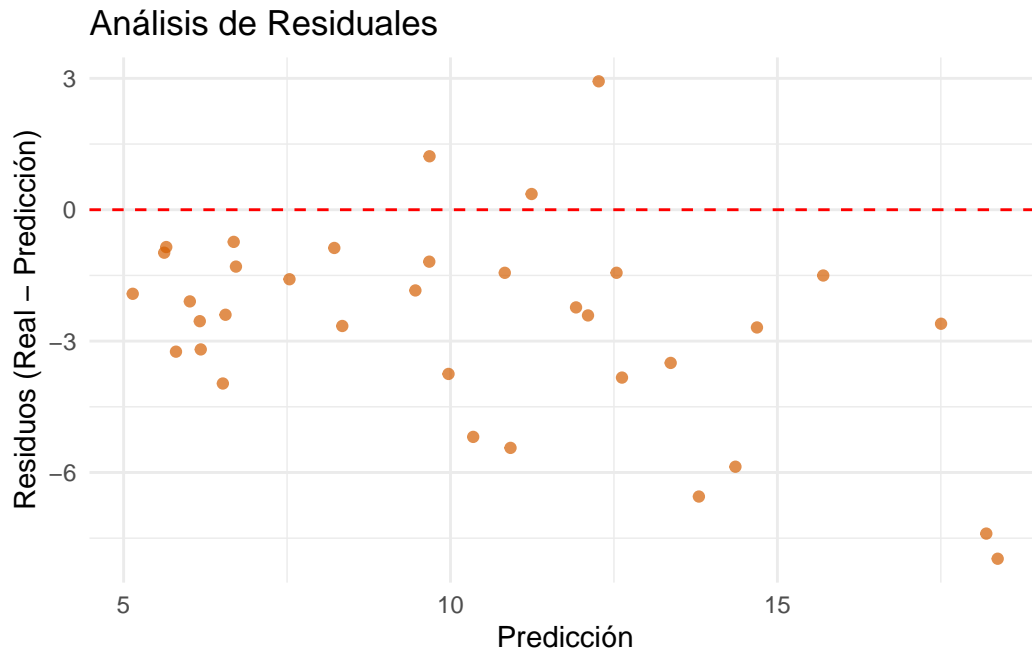
```
xg_serie_tiempo(modelo = modelo_opti_guardia, lista_datos = lista_guardia, nombre_fecha = "f
```

### Comparación entre valores reales y predicciones





```
xg_grafico_residuos(modelo_opti_guardia, lista_guardia, opti = TRUE)
```



## 4 Notas

Estuve viendo otros modelos como LightGBM y ExtraTrees que tal vez se podrían programar para ver sus resultados, intentaré para inicios del semestre haber cambiado el código del modelo de Redes Neuronales con otro framework, por ejemplo PyTorch y ver si se puede mejorar algo pues los resultados son bastante malos.

## 5 Anexos

Link del Repositorio : [https://github.com/AndreyPrado/Pronostico\\_Caudales\\_Extremos](https://github.com/AndreyPrado/Pronostico_Caudales_Extremos)