Modelitation con ARIMA

UDLAP
Econometria II
Daniel Strauss, ID: 186021
Christian McDonald, ID: 165650
Supervision: Daniela Toto
24.11.2024

Introduction

Objetivo del trabajo: Explorar el comportamiento de series temporales usando modelos ARIMA para pronósticos.

ARIMA: Modelo estadístico utilizado en economía y finanzas, adecuado para datos secuenciales con patrones de tendencia, estacionalidad y autocorrelación.

Aplicación: Optimizar la gestión de recursos digitales como servidores web, clave en el contexto tecnológico actual.

Datos utilizados: Accesos a servidores web en intervalos regulares para identificar patrones temporales complejos.

Proceso del estudio:

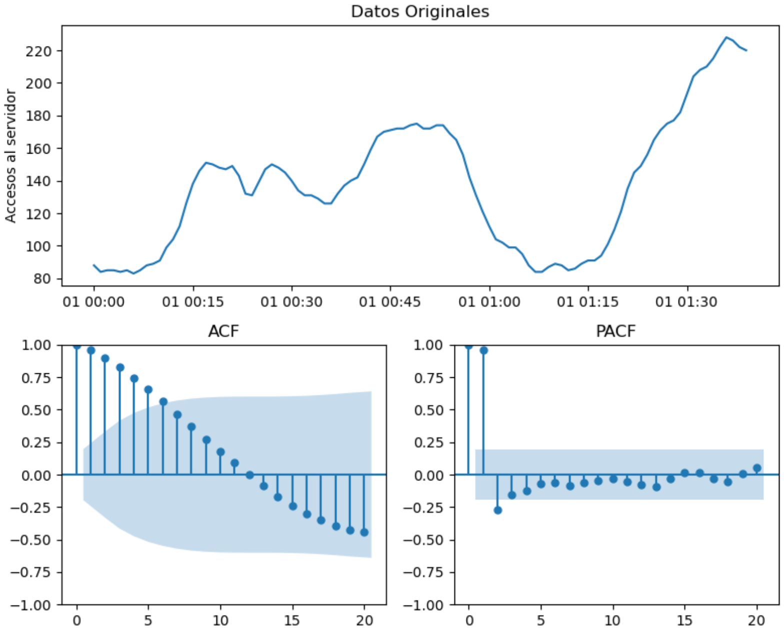
Limpieza y transformación de datos.

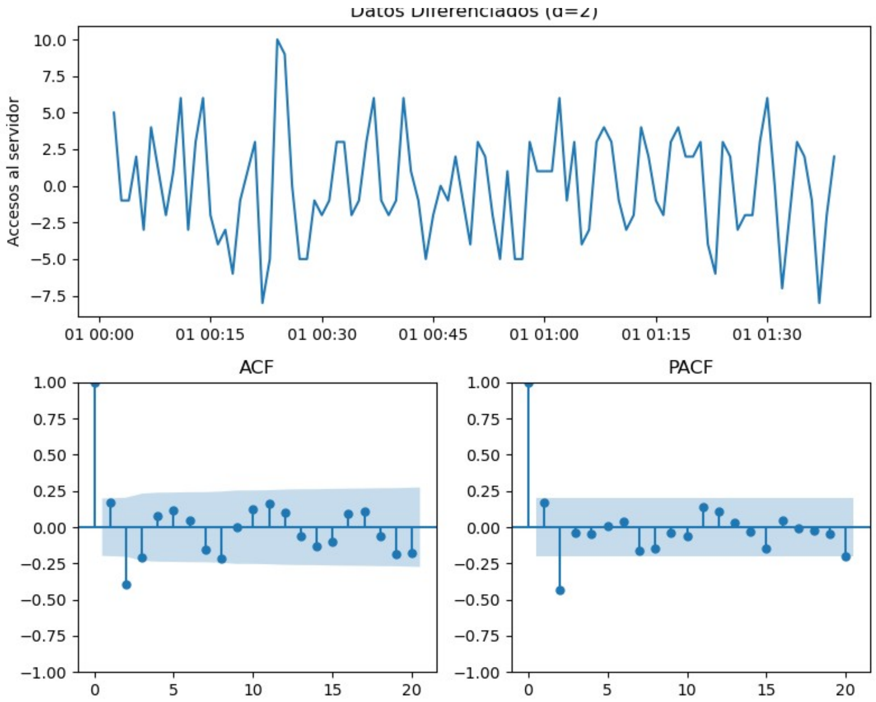
Ajuste del modelo ARIMA adecuado.

Evaluación de pronósticos y cumplimiento de supuestos necesarios.

Objetivo final: Optimizar recursos tecnológicos y demostrar la aplicabilidad de ARIMA en entornos digitales.

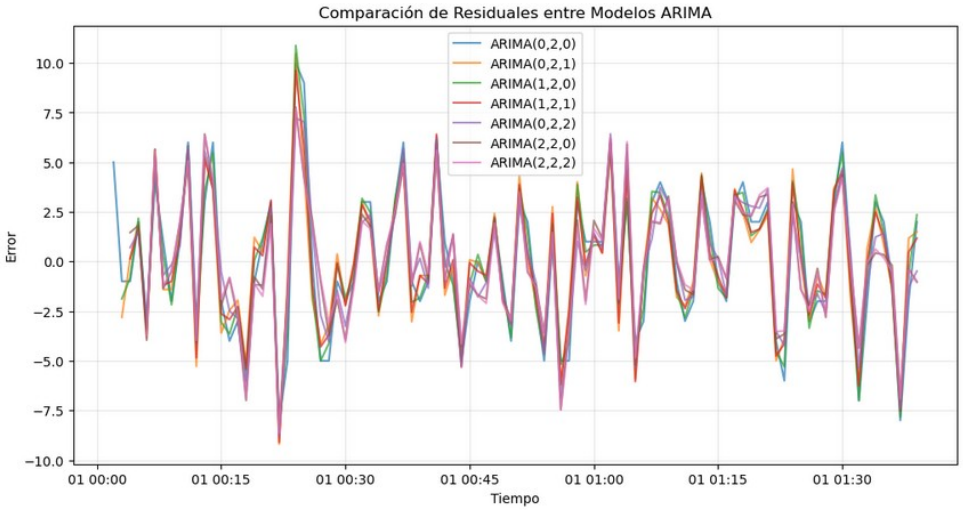
Preparacion de los Datos






```
Después de 1 diferenciación(es), el p-valor de ADF es: 0.07026846015272693
Después de 2 diferenciación(es), el p-valor de ADF es: 2.843428755547158e-17
```

Fit ARIMA Modelos




```python
from statsmodels.tsa.arima.model import ARIMA  # Importar el modelo ARIMA
```

```python
# Iterar sobre los modelos sugeridos y ajustar cada uno
for p, d, q in suggested_models:
    # Ajustar el modelo ARIMA con los parámetros actuales (p, d, q)
    model = ARIMA(data, order=(p, d, q))
    fitted = model.fit()
```

Verificar los Modelos, 8 Supuestos

Supuestos sobres los residuales

```python
# 1. Prueba de media cercana a 0 (t-test)
p_value = ttest_1samp(residuals, 0).pvalue
results['mean_close_to_0'] = p_value > alpha  # La media está cerca de 0 si p-valor > alpha
print(f"{model_name}: Media de residuales = {np.mean(residuals):.4f}, p-valor = {p_value:.4f}")

# 2. Prueba de varianza constante (homocedasticidad) usando Breusch-Pagan
_, pvalue, _, _ = het_breuschpagan(residuals, sm.add_constant(np.arange(len(residuals))))
results['constant_variance'] = pvalue > alpha  # Pasa si p-valor > alpha
print(f"{model_name}: Homocedasticidad (p-valor de Breusch-Pagan) = {pvalue:.4f}")

# 3. Pruebas de normalidad (Shapiro-Wilk y Jarque-Bera)
_, shapiro_pvalue = shapiro(residuals)  # Prueba Shapiro-Wilk
jb_stat, jb_pvalue = jarque_bera(residuals)  # Prueba Jarque-Bera
results['normal_distribution'] = shapiro_pvalue > alpha and jb_pvalue > alpha  # Ambas deben pasar
print(f"{model_name}: Normalidad (p-valor Shapiro-Wilk) = {shapiro_pvalue:.4f}")
print(f"{model_name}: Normalidad (p-valor Jarque-Bera) = {jb_pvalue:.4f}")

# 4. Prueba de independencia usando Ljung-Box
lb_test = acorr_ljungbox(residuals, lags=[10], return_df=True)
pvalue_ljungbox = lb_test['lb_pvalue'].values[0]
results['independent_errors'] = pvalue_ljungbox > alpha  # Pasa si p-valor > alpha
print(f"{model_name}: Independencia (p-valor de Ljung-Box) = {pvalue_ljungbox:.4f}")
```

```
Resumen de las Pruebas de Suposiciones de Residuales:
ARIMA(0,2,0):
  Media cercana a 0: Pasa
  Varianza constante: Pasa
  Distribución normal: Pasa
  Errores independientes: No pasa
----------------------------------------
ARIMA(0,2,1):
  Media cercana a 0: Pasa
  Varianza constante: Pasa
  Distribución normal: Pasa
  Errores independientes: No pasa
----------------------------------------
ARIMA(1,2,0):
  Media cercana a 0: Pasa
  Varianza constante: Pasa
  Distribución normal: Pasa
  Errores independientes: No pasa
----------------------------------------
ARIMA(1,2,1):
  Media cercana a 0: Pasa
  Varianza constante: Pasa
  Distribución normal: Pasa
  Errores independientes: No pasa
----------------------------------------
ARIMA(0,2,2):
  Media cercana a 0: Pasa
  Varianza constante: Pasa
  Distribución normal: Pasa
  Errores independientes: Pasa
----------------------------------------
ARIMA(2,2,0):
  Media cercana a 0: Pasa
  Varianza constante: Pasa
  Distribución normal: Pasa
  Errores independientes: Pasa
----------------------------------------
ARIMA(2,2,2):
  Media cercana a 0: Pasa
  Varianza constante: Pasa
  Distribución normal: Pasa
  Errores independientes: Pasa
----------------------------------------
```

Verificar los Modelos, 8 Supuestos

Supuesto 5: Modelo Parsimonioso

```python
# Obtener los intervalos de confianza para los parámetros del modelo
conf_intervals = fitted_model.conf_int()
parameters = fitted_model.params  # Obtener los valores estimados de los parámetros
```

```
Resumen de Significancia de Parámetros de Modelos ARIMA Basado en Intervalos de Confianza:
======================================================
Modelos que Aprobaron (Intervalos de Confianza NO contienen cero):
['ARIMA(0,2,0)', 'ARIMA(0,2,1)', 'ARIMA(2,2,0)']

Modelos que Fallaron (Intervalos de Confianza contienen cero):
['ARIMA(1,2,0)', 'ARIMA(1,2,1)', 'ARIMA(0,2,2)', 'ARIMA(2,2,2)']
```

Supuesto 6: Los modelos definen las series y prueban si son estationarias e invertibles

```python
# Obtener los parámetros AR (Autoregresivo) del modelo
ar_params = [param for param in fitted_model.params.index if 'ar.L' in param]
ar_coefficients = [fitted_model.params[param] for param in ar_params]  # Extraer lo

# Calcular las raíces del polinomio AR: 1 - p1*x - p2*x^2 - ...
ar_roots = np.roots(([1] + [-coeff for coeff in ar_coefficients])[::-1])  # Negar l
```

```
Resumen de la Prueba de Admisibilidad Basada en Raíces de los Polinomios MA y AR:
==================================================
Modelos que Pasaron (Modelos Admisibles con Raíces Fuera del Círculo Unitario):
['ARIMA(0,2,0)', 'ARIMA(0,2,1)', 'ARIMA(1,2,0)', 'ARIMA(1,2,1)', 'ARIMA(0,2,2)', 'ARIMA(2,2,0)', 'ARIMA
(2,2,2)']

Modelos que Fallaron (Modelos No Admisibles con Raíces Dentro o en el Círculo Unitario):
[]
```

Verificar los Modelos, 8 Supuestos

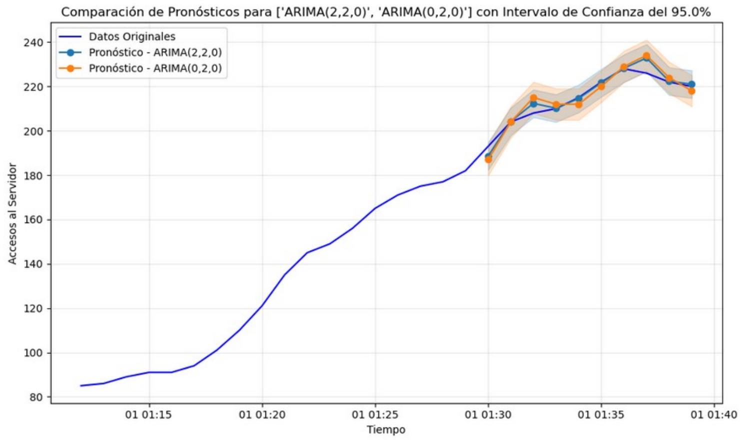
Supuesto 7,

Verifica si las correlaciones entre los parámetros estimados son pequeños.

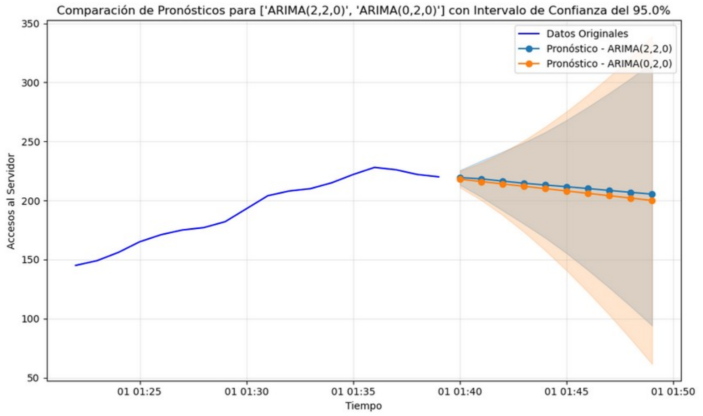
```python
# Extraer la matriz de varianza-covarianza de los parámetros estimados
cov_matrix = fitted_model.cov_params()  # Matriz de covarianza de los parámetros

# Convertir la matriz de covarianza a una matriz de correlación
correlation_matrix = cov2corr(cov_matrix)
```

```
Matriz de Correlación de Parámetros para ARIMA(0,2,0):
        sigma2
sigma2     1.0

Matriz de Correlación de Parámetros para ARIMA(0,2,1):
           ma.L1    sigma2
ma.L1   1.000000  0.045302
sigma2  0.045302  1.000000

Matriz de Correlación de Parámetros para ARIMA(1,2,0):
          ar.L1   sigma2
ar.L1   1.00000  0.12246
sigma2  0.12246  1.00000

Matriz de Correlación de Parámetros para ARIMA(1,2,1):
           ar.L1     ma.L1    sigma2
ar.L1   1.000000 -0.882550  0.084942
ma.L1  -0.882550  1.000000 -0.043799
sigma2  0.084942 -0.043799  1.000000

Matriz de Correlación de Parámetros para ARIMA(0,2,2):
           ma.L1     ma.L2    sigma2
ma.L1   1.000000  0.318789  0.135840
ma.L2   0.318789  1.000000 -0.095333
sigma2  0.135840 -0.095333  1.000000

Matriz de Correlación de Parámetros para ARIMA(2,2,0):
           ar.L1     ar.L2    sigma2
ar.L1   1.000000 -0.154842  0.235739
ar.L2  -0.154842  1.000000 -0.038835
sigma2  0.235739 -0.038835  1.000000

Matriz de Correlación de Parámetros para ARIMA(2,2,2):
           ar.L1     ar.L2     ma.L1     ma.L2    sigma2
ar.L1   1.000000 -0.141192 -0.900061 -0.092983  0.230904
ar.L2  -0.141192  1.000000  0.125011 -0.870712  0.121231
ma.L1  -0.900061  0.125011  1.000000  0.073469 -0.144525
ma.L2  -0.092983 -0.870712  0.073469  1.000000 -0.216743
sigma2  0.230904  0.121231 -0.144525 -0.216743  1.000000
```

Predictiones




```
# Realizar pronósticos
forecast = fitted_model.get_prediction(start=start, end=end)
forecast_mean = forecast.predicted_mean  # Valores pronosticados
forecast_ci = forecast.conf_int(alpha=1 - confidence_level)  # Intervalos de c
```



Conclusion

Objetivo del estudio: Analizar y predecir patrones en series temporales de accesos a servidores web utilizando modelos ARIMA.

ARIMA: Herramienta poderosa para series temporales no estacionarias con tendencias, estacionalidades y autocorrelaciones.

Resultados:

Validación de la eficacia de ARIMA para capturar patrones en accesos a servidores.

Identificación de los modelos ARIMA más adecuados mediante pruebas de estacionariedad.

Desafíos: Alta variabilidad y naturaleza no estacionaria de los datos de accesos.

Método:

Transformaciones adecuadas y ajustes de modelos ARIMA con AIC y BIC.

Modelos seleccionados proporcionan descripciones precisas y pronósticos confiables.

Validación: Análisis de residuales mostró cumplimiento de los supuestos de independencia, normalidad y homocedasticidad.

Pronósticos: Comparación de pronósticos con valores reales mostró buen desempeño en la predicción de la demanda de acceso a servidores.

Contribución: Metodología replicable aplicable a otros contextos tecnológicos con series temporales similares (e-commerce, finanzas, etc.).

Conclusión: Los modelos ARIMA son eficientes para analizar y predecir series temporales complejas, ofreciendo una base para futuras investigaciones y optimización de recursos tecnológico

Referencias

- Dataset "WWWUsage": Makridakis, S., Wheelwright, S. C. and Hyndman, R. J. (1998) Forecasting: Methods and Applications. Wiley