Inferència Estadística Actividad 4

Yolanda Colom Torrens

Introducción

Durante este problema se desarrollarán de manera distinta el informe, es decir se ira resolviendo paso a paso por Python como manipular el dataframe y los datos según lo que estamos buscando resolver, además de ir explicando cada método en particular y sus utilidades.

Problema 1

Primero debemos crear el dataframe y la tabla que deseamos evaluar para una serie temporal del tipo III

In [3]:
```
import pandas as pd
import numpy as np
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error


data = {
    't': [2003.2, 2004.1, 2004.2, 2005.1, 2005.2, 2006.1, 2006.2, 2007.1, 2007.2, 2008.1, 2008.2, 2009.1, 2009.2],
    'Yt': [13.09, 17.62, 20.05, 12.76, 18.64, 15.95, 21.32, 30.61, 30.86, 25.8, 30.74, 29.66, 33.67]
}

df = pd.DataFrame(data)
print(df)
```

```
         t     Yt
0   2003.2  13.09
1   2004.1  17.62
2   2004.2  20.05
3   2005.1  12.76
4   2005.2  18.64
5   2006.1  15.95
6   2006.2  21.32
7   2007.1  30.61
8   2007.2  30.86
9   2008.1  25.80
10  2008.2  30.74
11  2009.1  29.66
12  2009.2  33.67
```

Luego deberemos aplicar el uso de AEH, Dobles medias móviles y tendencia lineal.

-AEH(modelo de suavizamiento exponencial aditivo Holt-Winters), se utiliza para predecir series temporales. Para argumenta que los valores futuros esta relacionados con una combinación de los de su nivel actual, tendencia y el componente estacional. Para este modelo tenemos dos variables importantes alfa que controla la influencia del valor actual y gamma el cual controla la tendencia.

```
# AEH con alfa=0.3, gamma=0.6
model1 = ExponentialSmoothing(np.asarray(df['Yt'][:-3]), seasonal_periods=2, trend='add', seasonal='add')
model1 = model1.fit(smoothing_level=0.3, smoothing_slope=0.6)

# AEH con alfa=0.7, gamma=0.4
model2 = ExponentialSmoothing(np.asarray(df['Yt'][:-3]), seasonal_periods=2, trend='add', seasonal='add')
model2 = model2.fit(smoothing_level=0.7, smoothing_slope=0.4)

df['AEH_0.3_0.6'] = np.append(model1.fittedvalues, model1.forecast(3))
df['AEH_0.7_0.4'] = np.append(model2.fittedvalues, model2.forecast(3))

# Evaluamos cuál modelo es mejor
mse1 = mean_squared_error(df['Yt'][:-3], df['AEH_0.3_0.6'][:-3])
mse2 = mean_squared_error(df['Yt'][:-3], df['AEH_0.7_0.4'][:-3])

print(f'MSE AEH(0.3, 0.6): {mse1}')
print(f'MSE AEH(0.7, 0.4): {mse2}')
```

```
MSE AEH(0.3, 0.6): 23.549096218546143
MSE AEH(0.7, 0.4): 28.04581349869296
```

-EL método de dobles medias móviles: en esta técnica la predicción es simple y lo hace a partir del cálculo de promedios de dos niveles, al largo plazo y al corto plazo. Esto se realiza al extrapolar las diferencias de los valores futuros de ambos niveles. Para esto la variable K determinara el porte.

```
import pandas as pd
import numpy as np
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error


data = {
    't': [2003.2, 2004.1, 2004.2, 2005.1, 2005.2, 2006.1, 2006.2, 2007.1, 2007.2, 2008.1, 2008.2, 2009.1, 2009.2],
    'Yt': [13.09, 17.62, 20.05, 12.76, 18.64, 15.95, 21.32, 30.61, 30.86, 25.8, 30.74, 29.66, 33.67]
}

df = pd.DataFrame(data)

# Doble media móvil con k=4
df['SMA_4'] = df['Yt'].rolling(window=4).mean()

# Doble media móvil con k=2
df['SMA_2'] = df['Yt'].rolling(window=2).mean()

# Evaluamos cuál método es mejor
mse3 = mean_squared_error(df['Yt'][3:-3], df['SMA_4'][3:-3])
mse4 = mean_squared_error(df['Yt'][1:-3], df['SMA_2'][1:-3])

print(f'MSESMA(4): {mse3}')
print(f'MSE SMA(2): {mse4}')
```

```
MSESMA(4): 21.465456250000006
MSE SMA(2): 7.2829861111111125
```

Por último el método de tendencia lineal, este método funciona bajo la hipótesis de que la serie temporal sigue una tendencia lineal. Esta es muy utilizada en regresiones lineales extrapolando los valores futuros de la función, es necesario el cálculo de coeficientes de esta tendencia.

```
import pandas as pd
import numpy as np
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error


data = {
    't': [2003.2, 2004.1, 2004.2, 2005.1, 2005.2, 2006.1, 2006.2, 2007.1, 2007.2, 2008.1, 2008.2, 2009.1, 2009.2],
    'Yt': [13.09, 17.62, 20.05, 12.76, 18.64, 15.95, 21.32, 30.61, 30.86, 25.8, 30.74, 29.66, 33.67]
}

df = pd.DataFrame(data)

# Ajustar una línea a los datos
z = np.polyfit(df['t'][:-3], df['Yt'][:-3], 1)
p = np.poly1d(z)

df['trend_line'] = p(df['t'])

# Evaluar la precisión
mse5 = mean_squared_error(df['Yt'][:-3], df['trend_line'][:-3])

print(f'MSE Trend Line: {mse5}')
```

```
MSE Trend Line: 16.001139632235954
```

En este caso, el método de las dobles medias móviles con amplitud k=2 tiene el MSE más bajo (7.2829861111111125), por lo que se prefiere utilizar este para este caso ya que el método es el mejor de los que se han probado en estos datos.