Universidad Autónoma de Querétaro

Facultad de Ingeniería
División de Investigación y Posgrado

Práctica 2

Imputación de datos tabulares

Maestría en Ciencias en Inteligencia Artificial
Optativa de especialidad IV - Machine Learning

Aldo Cervantes Marquez
Expediente: 262775
Profesor: Dr. Marco Antonio Aceves Fernández

Santiago de Querétaro, Querétaro, México
Semestre 2023-1
03 de Marzo de 2023

Índice

1. Objetivo **1**

2. Introducción **1**

3. Marco Teórico **1**

3.1. Métodos de imputación para datos categóricos 1
3.1.1. Moda 1
3.1.2. Aleatorio 2
3.1.3. Hot-Deck 2
3.2. Métodos de imputación para datos cuantitativos no dependientes en el tiempo 2
3.2.1. Media 2
3.2.2. Mediana 2
3.3. Métricas de evaluación para datos cuantitativos y cualitativos no dependientes del tiempo 2
3.4. Métodos de imputación para datos cuantitativos dependientes en el tiempo 3
3.4.1. Medias Móviles 3
3.4.2. Media Móvil Ponderada 3
3.4.3. Método de Markov de primer orden 3
3.4.4. Método de Markov de Segundo Orden 4
3.5. Métricas de evaluación para datos continuos en series de tiempo 5
3.5.1. RMSE 5
3.5.2. MAE 5
3.5.3. MAD 5
3.5.4. Error Relativo Porcentual 5

4. Materiales y Métodos **5**

4.1. Materiales 5
4.1.1. Base de datos 5
4.1.2. Librerías y entorno de desarrollo 6
4.2. Metodología 6

5. Pseudocódigo **7**

6. Resultados **7**

6.1. Datos Categóricos 7
6.2. Datos cuantitativos no dependientes del tiempo 8
6.3. Datos de serie de tiempo 9
6.3.1. Completando serie de datos faltantes 9
6.3.2. Imputando datos aleatorios 11

7. Conclusiones **11**

Referencias **11**

8. Código Documentado **12**

1. Objetivo

Esta práctica tiene como principal objetivo el de obtener una base de datos tabular con datos faltantes y realizar una imputación de los mismos, mediante el uso de métodos de imputación para cada tipo de dato, comparando métodos para observar que tipo de método se adecua mejor a los datos en función de sus características.

2. Introducción

La presente práctica consiste en realizar imputación de datos en datos categóricos y datos continuos (series de tiempo). Aplicar métodos de imputación y comparar los resultados obtenidos en cada método.

Por lo que los principales datos a tratar son:

- Datos categóricos.
- Datos cuantitativos
- Datos continuos (series de tiempo).

Para el uso de estos datos, se deberán conocer los métodos adecuados para cada tipo de imputación según el tipo de dato. Por lo que también será de interés variar la cantidad de datos faltantes, hiperparámetros de los algoritmos, etc.

3. Marco Teórico

Para la realización de la práctica fueron necesarios los siguientes conceptos, los cuales fueron obtenidos principalmente de [1].

3.1. Métodos de imputación para datos categóricos

Para este tipo de datos, los cuales no tienen meramente un significado numérico, se les puede aplicar principalmente las siguientes técnicas de imputación:

3.1.1. Moda

Básicamente consiste en calcular el valor más repetido del conjunto de datos y se puede definir como $\hat{M}_o$. Aplicando la siguiente fórmula

$$\hat{M}_o = d_i[\text{máx}(f_{d_i})] \in d, \qquad d[faltante] = \hat{M}_o \tag{1}$$

3.1.2. Aleatorio

Este método consiste en tomar un valor en el rango de las categorías y colocarlo en el valor imputado, cuando se encuentre otro valor vacío realizar el mismo procedimiento, este tipo de selección puede ser con algún tipo de distribución (uniforme, Gaussiana, exponencial, etc.). Se puede describir de manera general como:

$$m_A = rand(d_i \in d, PDF), \qquad d[faltante] = m_A \tag{2}$$

3.1.3. Hot-Deck

Este método consiste en encontrar el primer dato faltante e imputar el dato inmediato anterior no faltante, al encontrar otro dato faltante, se toma el siguiente dato no faltante y se reemplaza

$$h_d = d[faltante - 1], \qquad d[faltante] = h_d \tag{3}$$

3.2. Métodos de imputación para datos cuantitativos no dependientes en el tiempo

Para estos tipos de datos se pueden aplicar los métodos anteriores y se puede agregar un par de métodos basados en medidas de tendencia central, como son:

3.2.1. Media

En este método se ingresa la media aritmética de los datos, sustituyéndolo en lugar de los valores faltantes.

$$\bar{M} = \frac{\sum_{i=1}^{n} d_i}{n}, \qquad d[faltante] = \bar{m} \tag{4}$$

3.2.2. Mediana

Este método consiste en encontrar el valor del medio del conjunto de datos agrupados de menor a mayor o biceversa.

$$\tilde{M} = d[\tfrac{n}{2}], \qquad d[faltante] = \tilde{M} \tag{5}$$

3.3. Métricas de evaluación para datos cuantitativos y cualitativos no dependientes del tiempo

Para este tipo de datos, un histograma o barra de frecuencias y una gráfica de densidad de distribución son suficientes para observar como cambiaron los datos con respecto a los que se tenían anteriormente.

3.4. Métodos de imputación para datos cuantitativos dependientes en el tiempo

A estos métodos se agrega el método aleatorio mencionado anteriormente, donde se selecciona algún valor existente en la serie de tiempo para sustituir el valor faltante.

3.4.1. Medias Móviles

Debido a la dependencia de los datos y posible correlación entre los mismos, este método de imputación es bastante útil. Básicamente consiste en promediar un n número de valores anteriores [2].

$$\bar{M}M = \frac{\sum_{i=1}^{n} d_i}{n} \qquad d[daltante] = \bar{M}M \tag{6}$$

3.4.2. Media Móvil Ponderada

Este método es parecido al anterior, únicamente que cambia la división, pues mientras mas se aleja del dato faltante, menor peso tendrá, se puede expresar como.

$$\bar{M}M_p = \frac{\sum_{i=1}^{n} i*d_i}{\frac{n*(n+1)}{2}} \qquad d[daltante] = \bar{M}M_p \tag{7}$$

3.4.3. Método de Markov de primer orden

Este método consiste en el uso de cadenas de Markov (MCMC), las cuales consisten en una matriz de transición M_t y una matriz de estado s_t. La primera describe la probabilidad de cambios de estados según sea el caso, tiene la forma (nxm) donde n es la cantidad de casos que se tienen y m es la cantidad de estados que existen (es muy común que $n = m$). La matriz de estado muestra las probabilidades que tiene cada estado de ser seleccionado, por lo que tanto la suma de las columnas de cada renglón debe sumar 1 [3, 4]. Entonces se puede escribir de la siguiente manera :

$$m_t = \begin{bmatrix} p(1,1) & p(1,2) & p(1,3) & \cdots & p(1,m) \\ p(2,1) & p(2,2) & p(2,3) & \cdots & p(1,m) \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ p(n,1) & p(n,2) & p(n,3) & \cdots & p(n,m) \end{bmatrix}$$

(8)

donde las probabilidades se calculan mediante la frecuencia de la combinación de las clases:

$$p(n,m) = \frac{f(n,m)}{\sum f(m_t(:,m))} \tag{9}$$

Por otro lado se tiene que la matriz de transición con la forma:

$$s_t = \begin{bmatrix} p(s_1) & p(s_2) & p(s_3) & \cdots & p(s_n) \end{bmatrix} \tag{10}$$

Generalmente se toma como 1 en el estado inmediato al anterior de la matriz de estados.

Finalmente para obtener la imputación se debe calcular:

$$s_{t+1} = s_t \times m_t \tag{11}$$

Cabe destacar que las matrices deben actualizarse con los resultados obtenidos y actualizando la matriz de transición para conocer el nuevo panorama de datos y distribución de los mismos.

Finalmente cabe destacar que existen varios métodos de decodificación de los datos para este caso de datos continuos. Al obtener la nueva matriz de estado, se puede codificar de 2 maneras: obtener el de mayor probabilidad y elegir un valor (aleatorio, medio, etc.) dentro del estado, es decir aplicar una función de decodificación δ de la siguiente manera $y_{t+1} = \delta(\text{máx}(s_{t+1}))$, la segunda manera es ponderando los valores de la matriz de estados y sumarlos para obtener la salida mediante una función ϕ de la siguiente manera $y_{t+1} = \phi(\sum s_{t+1} * \delta)$ en donde se pondera la función decodificadora δ mencionada anteriormente.

3.4.4. Método de Markov de Segundo Orden

Este método es parecido al de primer orden, sin embargo se basa principalmente en la diferencia de que no necesariamente se requiere la matriz de estado, pues este método consiste en añadir un estado anterior a la matriz de transición para tener una matriz de ($nxmxm$) donde se observa la probabilidad de la combinación de 3 estados hacia atrás (anterior, actual y siguiente). Obteniendo una matriz de transición como se muestra en la Figura 1.

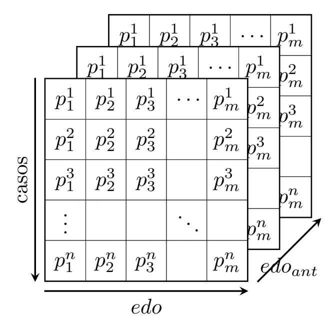


Figura 1: Matriz multidimensional para orden mayor a 1.

La manera más sencilla de decodificar los resultados es mediante el uso de la ubicación de los dos estados anteriores, pues conociendo esa combinación, tendremos una probabilidad para el estado siguiente (el que es de interés mediante las coordenadas), siendo toda la fila restante de la combinación, obteniendo la máxima probabilidad y aplicando las funciones *alpha* y *phi*.

3.5. Métricas de evaluación para datos continuos en series de tiempo

3.5.1. RMSE

El error medio cuadrático es definido como:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{12}$$

Por lo que al ser menor, se observará una mejor similitud de los datos reales con los imputados.

3.5.2. MAE

El error medio absoluto se define como:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{13}$$

De igual manera mientras menor sea, es mejor similitud.

3.5.3. MAD

Desviación absoluta de la media, permite conocer la variación que tenemos entre nuestros datos imputados. Una mayor variación no necesariamente significa una mejor similitud, pero si puede ayudar a explicar que tan capaz es el método de tratar en distintos rangos de operación. Por lo que su valor adecuado dependerá de la aplicación.

$$MAD = \frac{\sum_{i=1}^{n}|y_i - \bar{y}|}{n} \tag{14}$$

3.5.4. Error Relativo Porcentual

Esta métrica se ocupa cuando son datos individuales, con el fin de observar su desviación del valor verdadero.

$$e_{rp} = \frac{|\hat{y} - y|}{|\hat{y}|} * 100\,\% \tag{15}$$

4. Materiales y Métodos

4.1. Materiales

4.1.1. Base de datos

La base de datos elegida para los datos no dependientes del tiempo fue obtenida de la página de kaggle, la cual consta de datos de clientes de una tienda departamental durante los años de 2017-2020, enfocando la información a 4 pilares principales:

1. Perfil del cliente.

2. Preferencias de productos.
3. Éxito/fracaso de campañas publicitarias.
4. Frecuencia de consumos.

Sin embargo en este caso solo se extrajo el atributo de edades y de estado marital (2205 instancias).

Por otro lado, la serie de tiempo fue obtenida de una base de datos de Yahoo finance, la cual consta del valor de *Nasdaq composite* a lo largo de un año (sin contar algunos dias festivos y no laborales) Figura 2.

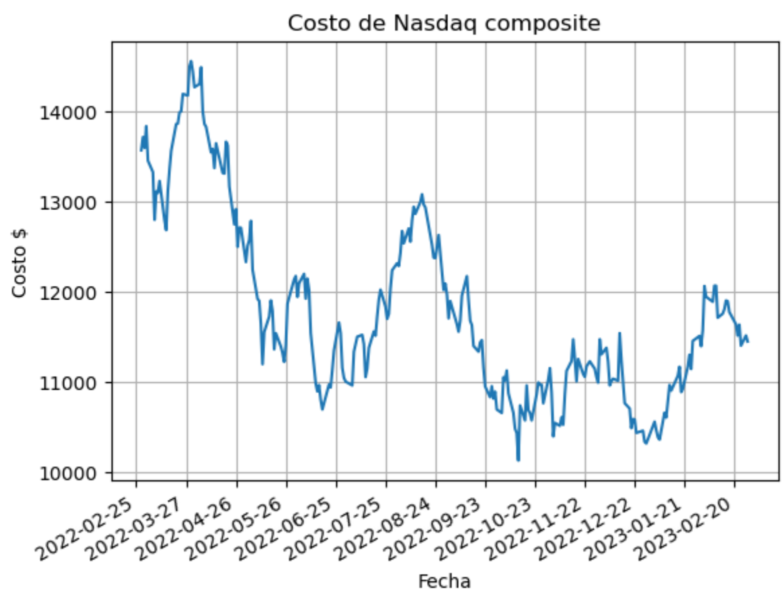


Figura 2: Serie de tiempo costo de acción Nasdaq composite.

4.1.2. Librerías y entorno de desarrollo

El análisis de los datos se llevará a cabo en el lenguaje de programación Python dentro del entorno de desarrollo de Jupyter Notebook. Ocupando las librerías matplotlib, seaborn, numpy y Pandas.

4.2. Metodología

La metodología consiste en la recopilación de las bases de datos, un estudio de los métodos de imputación, su desarrollo y su evaluación, variando algunos parámetros de los métodos y de la cantidad faltante de datos (véase Figura 3).

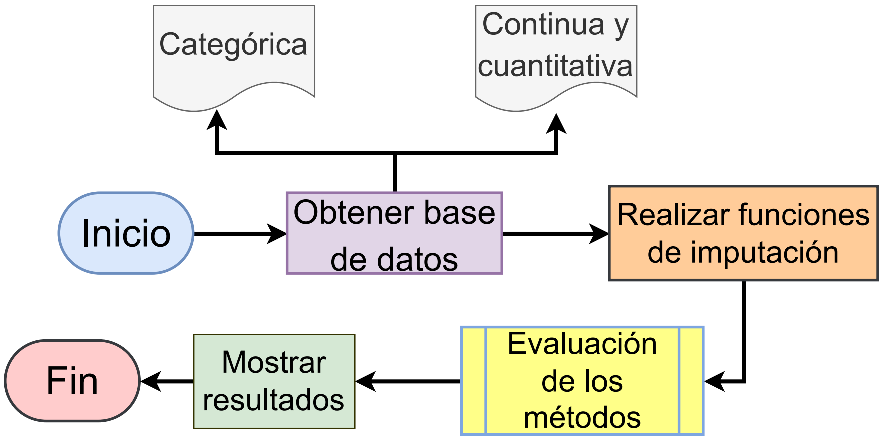


Figura 3: Metodología de la práctica.

5. Pseudocódigo

Algoritmo 1 Pseudocódigo imputación de datos categóricos y continuos.

Inicio
  Db
  Dbf ← Db*n_faltantes
  Función(media, mediana, moda, aleatorio)
  Función(media móvil, media móvil ponderada, markov1o1, markov2o1, markov2, aleatorio)
  Para método en cada **Función:**
    Dbf ← Función(método)
    Mostrar_Resultados(**Dbf**)
    Calcular_métricas(**Dbf,método**)
Fin

6. Resultados

6.1. Datos Categóricos

Para la sección de estatus marital recordemos la nomenclatura:

- Si es anonima = 0
- Si es divorciad@ = 1
- Si es casad@ = 2
- Si es solter@ = 3

- Si esta juntad@ = 4
- Si es viud@ =5
- Si se desconoce (dato faltante) = 6

Entonces, de la base se realizaron 4 pruebas en las que se iba haciendo una mayor cantidad de datos faltantes (10 %, 20 %, 50 % y 60 %). Observando sus distribuciones como se observa en la Figura 4.

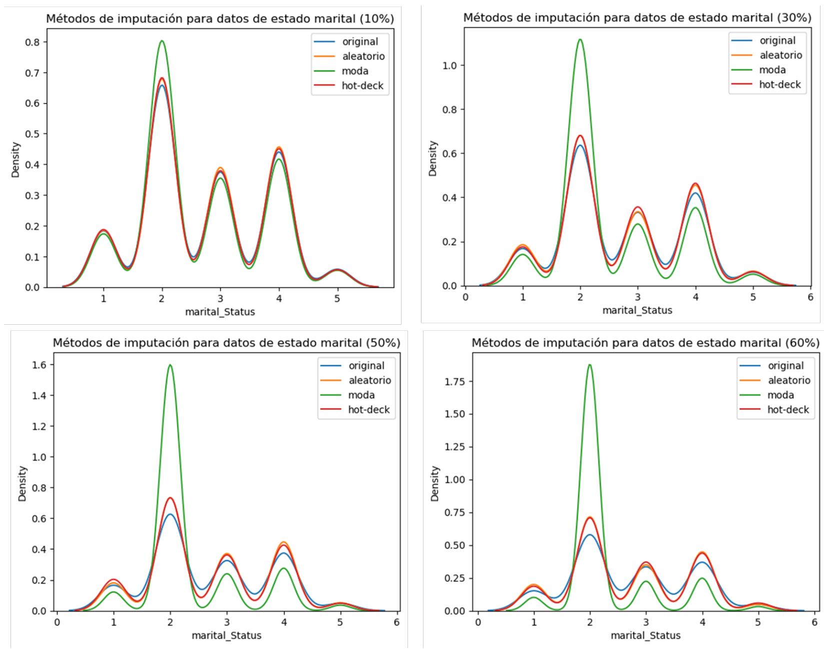


Figura 4: Distribución de imputaciones para diferentes proporciones de valores faltantes en estado marital.

Como se observa, el método de moda distorsiona cada vez más mientras se tienen menos datos, por lo que se va perdiendo la consistencia original de los datos. Por otro lado el método aleatorio y hot-deck no modifican de una manera tan significativa en comparacion con la moda, el modo en el que se distorsionan aunque se observe una tendencia a seguir distorsionando.

6.2. Datos cuantitativos no dependientes del tiempo

En el rango de las edades, se obtuvieron los siguientes resultados aplicando los mismos casos de cantidad de datos faltantes como se observa en la Figura 5.

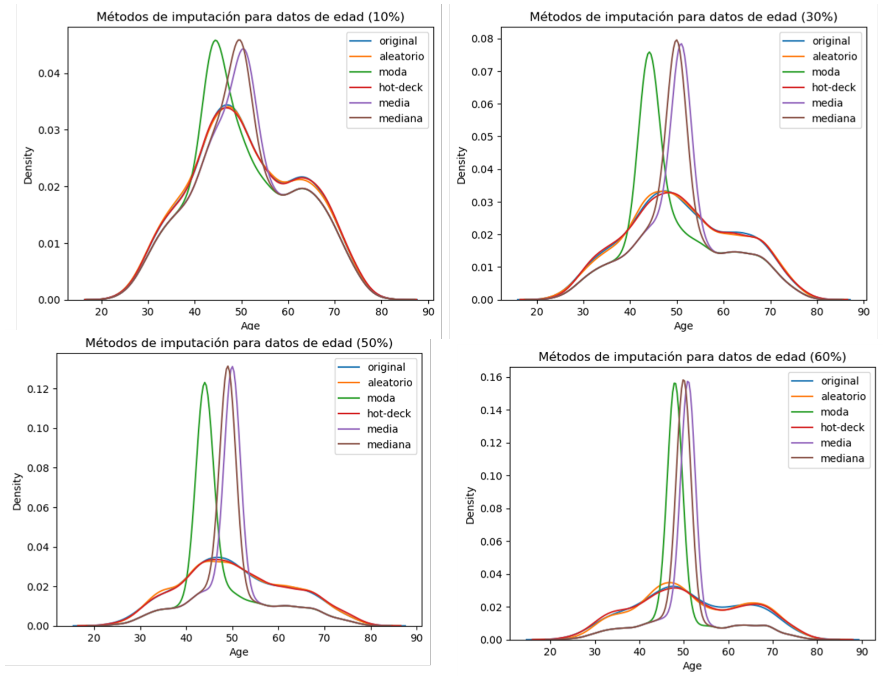


Figura 5: Distribución de imputaciones para diferentes proporciones de valores faltantes en edades.

De igual manera que con los datos del estatus marital, se observa como es que las imputaciones basadas en medidas de tendencia central, concentran toda la información en sus respectivos valores, por lo que no son una opción muy adecuada cuando se tienen muchos datos faltantes.

6.3. Datos de serie de tiempo

Para esta parte, se realizaron 2 pruebas, la primera consta en una serie de tiempo de 25 datos faltantes, con el fin de observar el comportamiento de la serie. La segunda consta de 5 datos aleatorios con el fin de observar su desempeño punto a punto y obtener sus errores.

6.3.1. Completando serie de datos faltantes

Se realizó una imputación de 25 días del año, comparando cada método, como se observa en la Figura 6.

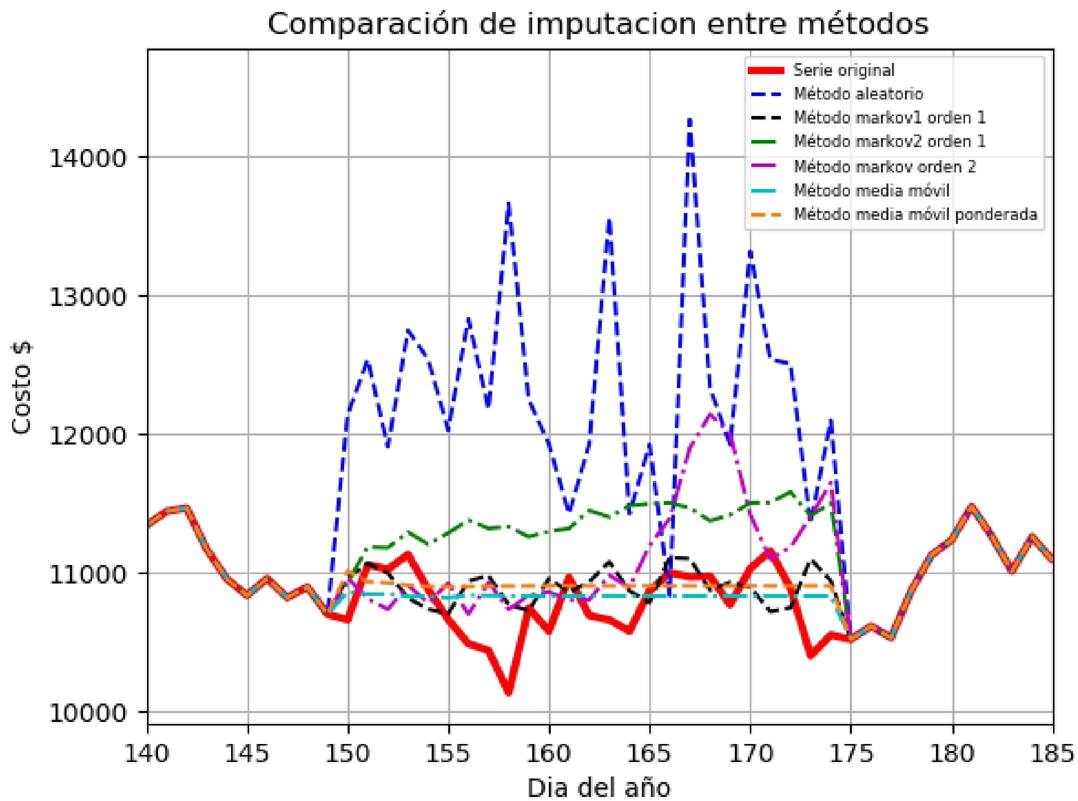


Figura 6: Distribución de imputaciones para diferentes proporciones de valores faltantes en edades.

Como se observa, el método aleatorio a simple vista es el menos semejante, los métodos basados en medias suavizan demasiado la salida casi a una linea recta, y los métodos de Markov logran predecir algunas de las oscilaciones pero no en la magnitud exacta. A continuación se muestra la evaluación de las métricas para cada método (véase Tabla 1).

Tabla 1: Desempeño de cada método de imputación.

Método		RMSE	MAE	MAD	MAD real
Estocástico	Aleatorio	1752.22	1570.033	580.74	214.54
Múltiple	Markov 1 orden 1	317.95	252.36	107.69	
	Markov 2 orden 1	655.1019	591.511	111.20	
	Markov orden 2	559.17	436.53	335.19	
Determinista	Media movil	260.13	217.05	4.62	
	Media móvil ponderada	288.34	227.49	11.305	

A simple vista se podría decir que el método de media móvil y media ponderada (deterministas) son los mejores dadas las métricas, sin embargo se puede cuestionar un poco por el MAD, pues es demasiada baja su variabilidad, por lo que se observa la imputación totalmente suavizada, además de que la serie original tiene bastantes variaciones, lo que pierde su semejanza. Por lo que los métodos de Markov aunque no sean tan exactos, imputan valores con las mismas tendencias.

6.3.2. Imputando datos aleatorios

Para esta prueba, se tienen 5 valores aleatorios que deben ser imputados y se observa el error relativo porcentual, para observar la cantidad de desviación que hay (véase Tabla 2).

Tabla 2: Desempeño de cada método de imputación con 5 valores aleatorios.

	Método / e_{rp} %											
Valor verdadero	Aleatorio	e_{rp}	Mk1o1	e_{rp}	Mk2o1	e_{rp}	Mko2	e_{rp}	mm	e_{rp}	mmp	e_{rp}
11167.37988	13837.5898	23.9108008	11228.763	0.54966434	11296.2229	1.15374406	11485.0294	2.84444078	11493.2967	2.91847177	11481.3276	2.81129265
11542.24023	13837.5898	19.8865174	11957.6264	3.5988345	11677.8084	1.17453947	11779.7028	2.05733545	11127.2598	3.59531997	11080.3488	4.00174841
13623.7002	13716.7002	0.68263393	13576.235	0.34840153	13648.8512	0.18461223	13399.0376	1.64905696	13483.7417	1.02731632	13423.6829	1.46815693
12506.37012	12375.1504	1.04922311	12228.975	2.21803074	11965.307	4.32629999	11573.1371	7.46206167	12669.5617	1.30486758	12669.5414	1.30470527
11707.44043	12506.3701	6.82411917	11795.1251	0.74896563	11776.5988	0.59072194	11852.3356	1.23763358	12266.4616	4.77492208	12277.8909	4.87254606
Promedio	10.4706589 %		1.49277935 %		1.485983539 %		3.05010569 %		2.72417955 %		2.89168987 %	

Como se observa los métodos de imputación múltiple tuvieron un mejor desempeño en cuanto a puntos aislados. Por lo que se podría recomendar para este caso el método 1 y 2 de Markov.

7. Conclusiones

La imputación de datos es de gran importancia debido a la necesidad de llenar esos espacios con un dato lo mejor estimado posible. Por lo que estos métodos tienen sus ventajas y desventajas con respecto a cada caso, por lo que no significa que uno siempre sea mejor que el otro. Pero se observa que en el caso especifico de Nasdaq, se tenia mucha variación en los datos, lo que podía hacer que algunos métodos fallaran, pero todos en general pudieron realizar al menos una imputación bastante aceptable, tanto que no moviera mucho la distribución, como que coincidiera con las métricas. Se observó también que muchos de estos métodos son relativamente sencillos de implementar, por lo que son viables para poder ser usados en otro tipo de datos, siempre y cuando se realice un analisis semejante al presentado.

Referencias

[1] M. Fernández, *Inteligencia Artificial para Programadores con Prisa.* Amazon Digital Services LLC - KDP Print US, 2021.

[2] "Series de tiempo." http://www.estadistica.mat.uson.mx/Material/seriesdetiempo.pdf. (Accessed on 02/27/2023).

[3] IBM, "Método (imputación múltiple) - documentación de ibm." https://www.ibm.com/docs/es/spss-statistics/saas?topic=imputation-method-multiple. (Accessed on 03/02/2023).

[4] C. Acevedo, "Aplicación de cadenas de markov para el analisis y pronostico de series de tiempo." http://tangara.uis.edu.co/biblioweb/tesis/2011/141227.pdf, 2011. (Accessed on 03/02/2023).

Imputación de datos ACM

March 2, 2023

1 Imputación de datos

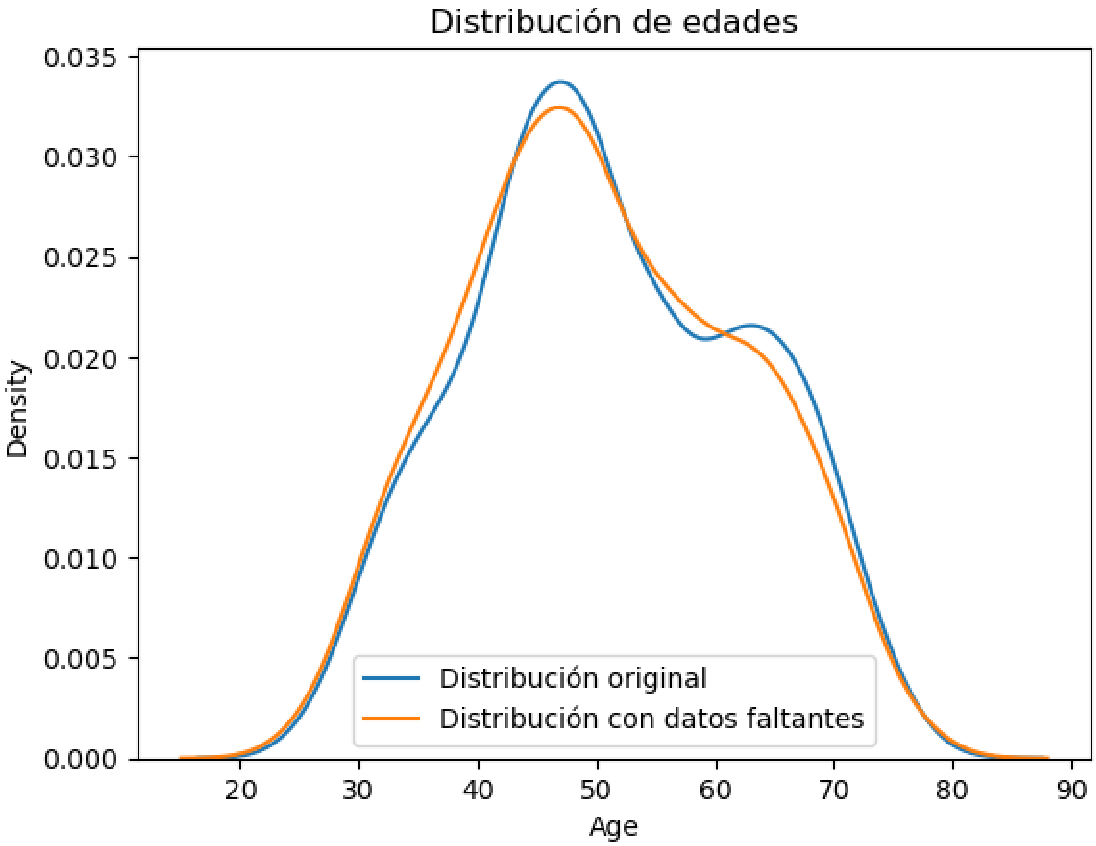
1.1 Numérico edades

hot-deck: consiste en elegir el valor más cercano (anterior) al dato faltante y a partir de ahi se va recorriendo uno a uno

[32]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
database=pd.read_csv("C:\\Users\\aldoa\\Machine Learning\\Practica 1 datos␣
↪tabulares\\Database_mejorada.csv")

# hacer base de datos faltantes
db1=database['Age'].copy()
#print(db1.shape[0])
porc_fal=0.6 # Cambiar cantidad de datos faltantes
n_faltantes=int(porc_fal*db1.shape[0])
falt=np.random.choice(db1.shape[0],size=(1,n_faltantes),replace=False)
#for a in range(n_faltantes):
db2=database['Age'].copy()
for a in range(n_faltantes):
    db2[falt[0,a]]=np.nan

print('valores faltantes',db2.isna().sum())

sns.kdeplot(database['Age']).set(title='Distribución de edades')
sns.kdeplot(db2)
plt.legend(['Distribución original','Distribución con datos faltantes'])
#sns.title('distribución original de edades')
```

```
valores faltantes 1323
```

[32]:
```
<matplotlib.legend.Legend at 0x1e281e6eca0>
```



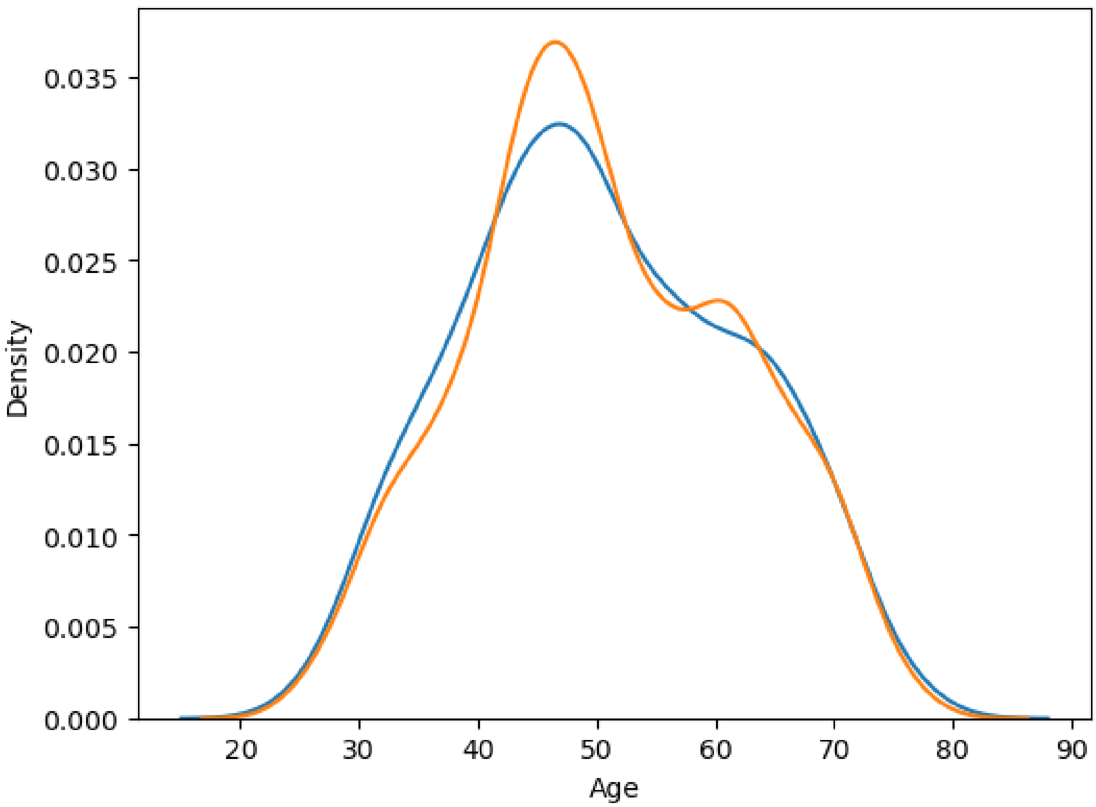
[33]:
```
def hot_deck(db):
    p=0
    for ab in range(db.shape[0]):
        if np.isnan(db[ab]):
            p=ab
            if p==0:
                db[0]=np.random.choice(db[~np.isnan(db)])
            else:
                break


    for xt in range(db.shape[0]):
        if np.isnan(db[xt]):
            db[xt]=db[p-1]
            p+=1
    return db

def i_al(db):
    rch=0
    for xt in range(db.shape[0]):
```

```
        if np.isnan(db[xt]):
            rch=np.random.choice(db[~np.isnan(db)])
            db[xt]=rch
    return db

# moda
def moda(db):
    moda=db.mode()[0]
    for xt in range(db.shape[0]):
        if np.isnan(db[xt]):
            db[xt]=moda
    return db

def media(db):
    med=int(db.mean())
    db=db.fillna(med)
    return db
def mediana(db):
    med=int(db.median())
    db=db.fillna(med)
    return db
```

[34]:
```
# Método aleatorio
db3=db2.copy()
dbrand=i_al(db3)
print(dbrand.isna().sum())
sns.kdeplot(db2)
sns.kdeplot(dbrand)
```

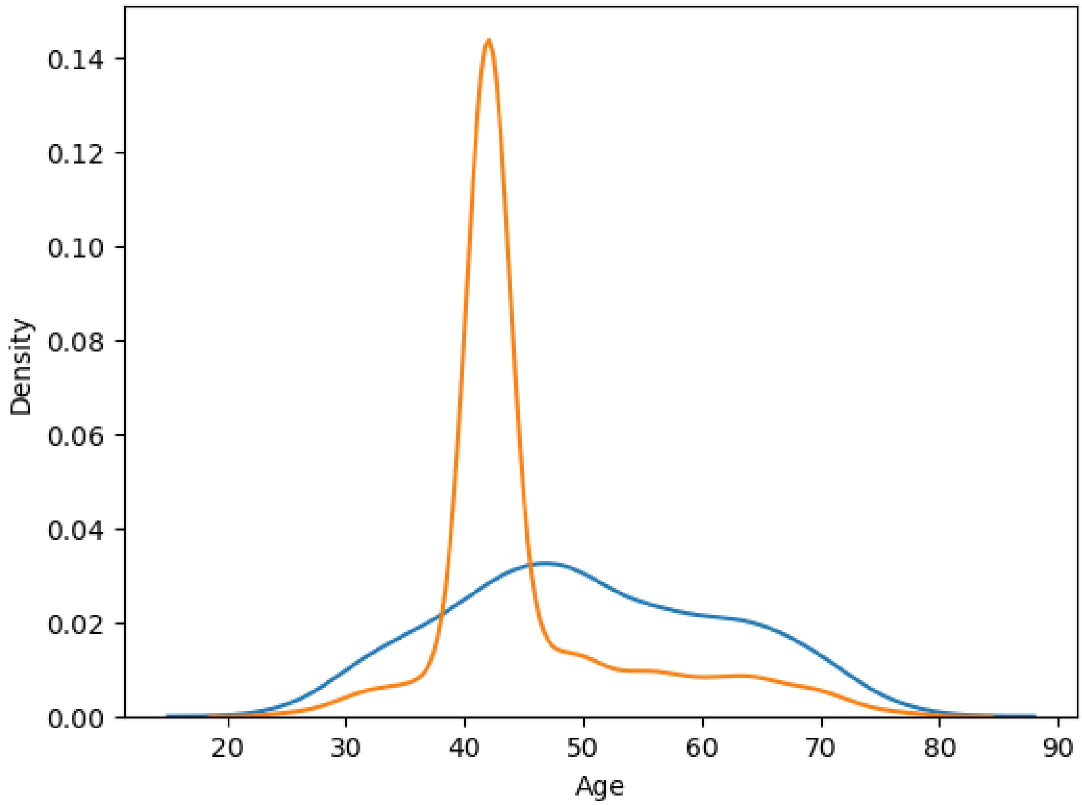
```
0
```

[34]:
```
<AxesSubplot:xlabel='Age', ylabel='Density'>
```



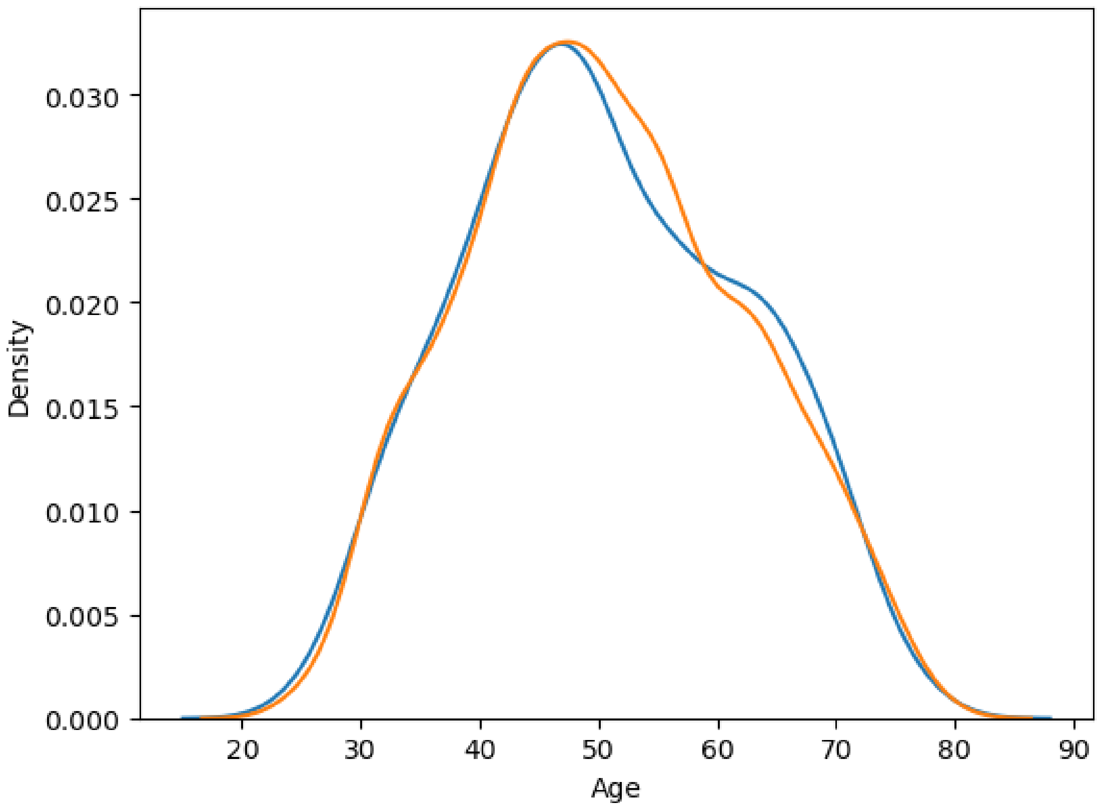
[35]:
```
db4=db2.copy()
dbmoda=moda(db4)
sns.kdeplot(db2)
sns.kdeplot(dbmoda)
print(dbmoda.isna().sum())
```

```
0
```



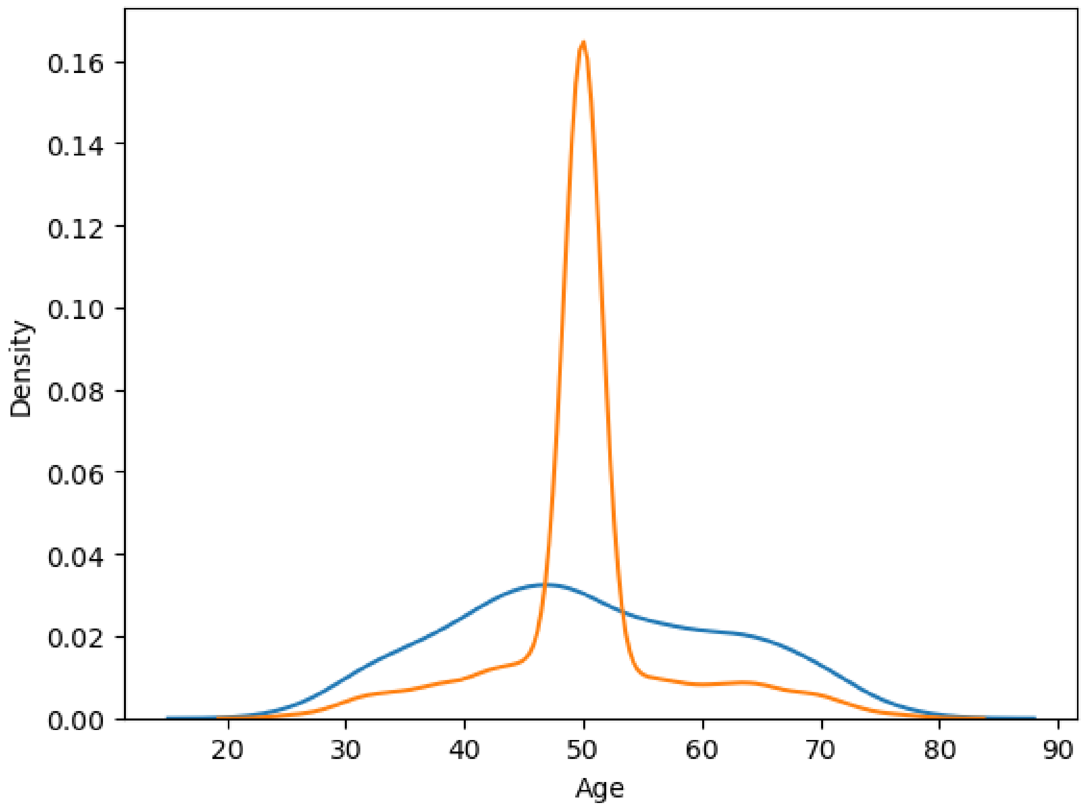
[36]:
```
db5=db2.copy()
dbhd=hot_deck(db5)
sns.kdeplot(db2)
sns.kdeplot(dbhd)
print(dbhd.isna().sum())
```

```
0
```



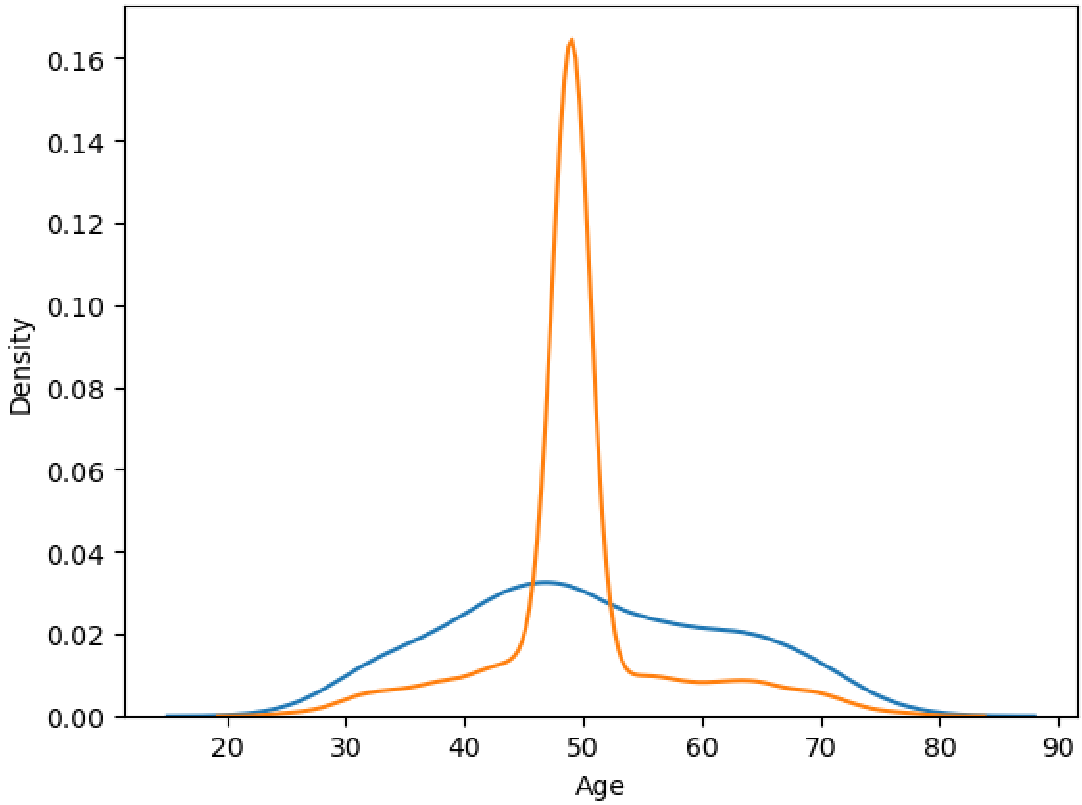
[37]:
```
# media
db6=db2.copy()
dbmed=media(db6)
sns.kdeplot(db2)
sns.kdeplot(dbmed)
print(dbmed.isna().sum())
```

```
0
```



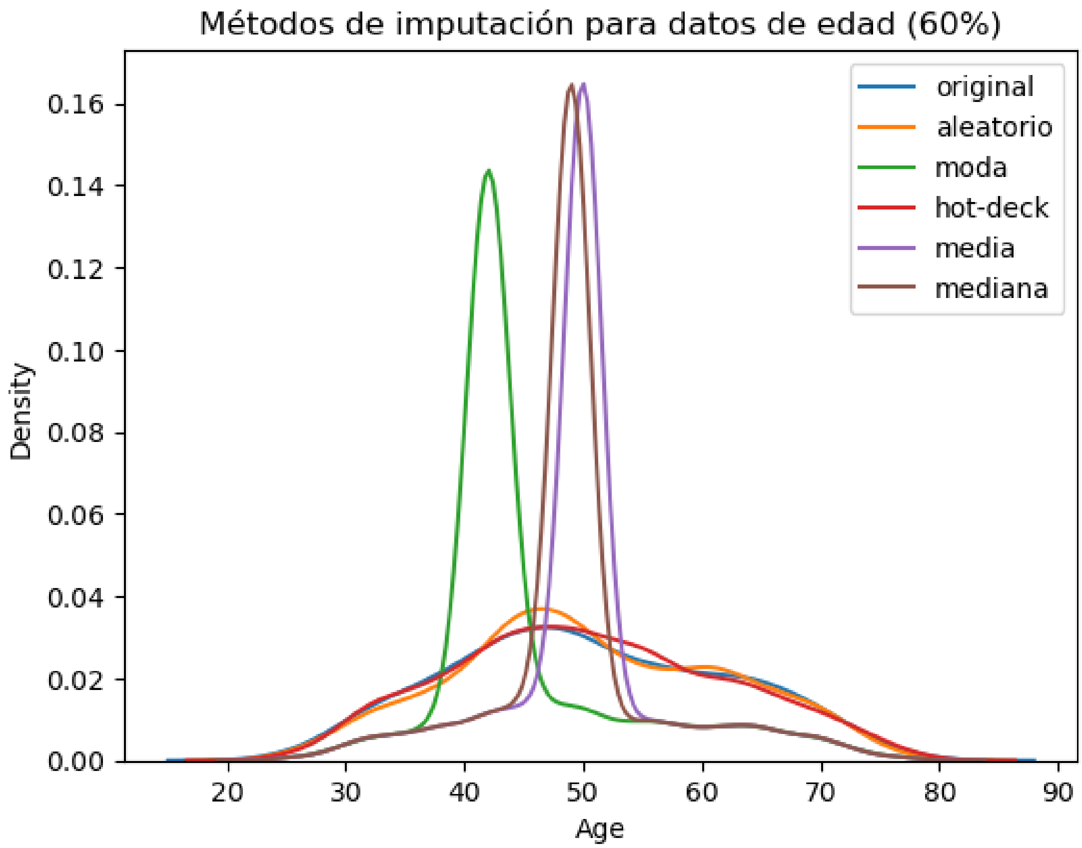
[38]:
```
# Mediana
db7=db2.copy()
dbmediana=mediana(db7)
sns.kdeplot(db2)
sns.kdeplot(dbmediana)
print(dbmediana.isna().sum())
```

0



[39]:
```
## Distribuciones
sns.kdeplot(db2).set(title='Métodos de imputación para datos de edad (60%)')
sns.kdeplot(dbrand)
sns.kdeplot(dbmoda)
sns.kdeplot(dbhd)
sns.kdeplot(dbmed)
sns.kdeplot(dbmediana)
plt.legend(['original','aleatorio','moda','hot-deck','media','mediana'])
```

[39]: <matplotlib.legend.Legend at 0x1e281889940>

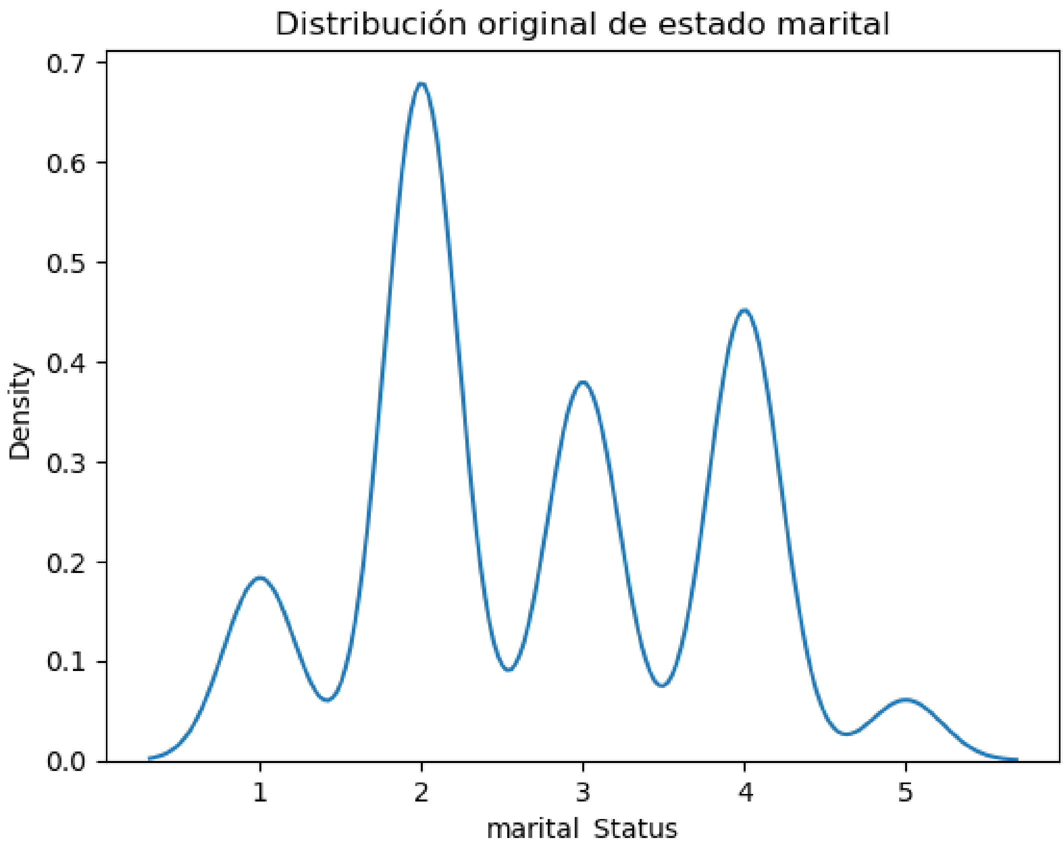


2 Para estatus marital

- Si es anonima = 0
- Si es divorciad@ = 1
- Si es casad@ = 2
- Si es solter@ = 3
- Si esta juntad@ = 4
- Si es viud@ =5
- Si se desconoce (dato faltante) = 6

[40]:
```
sns.kdeplot(database['marital_Status']).set(title='Distribución original de␣
↪estado marital')
```

[40]:
```
[Text(0.5, 1.0, 'Distribución original de estado marital')]
```



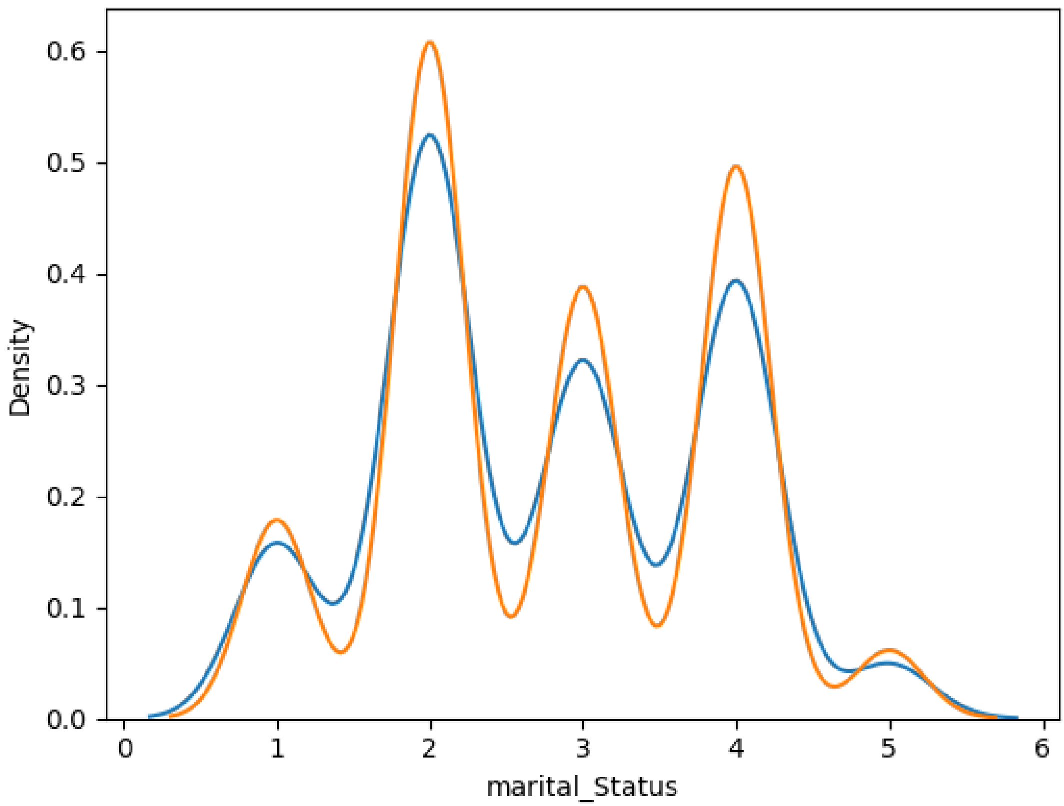
[41]:
```
# hacer base de datos faltantes
db1=database['marital_Status'].copy()
#print(db1.shape[0])
porc_fal=0.6
n_faltantes=int(porc_fal*db1.shape[0])
falt=np.random.choice(db1.shape[0],size=(1,n_faltantes),replace=False)
#for a in range(n_faltantes):
db2=database['marital_Status'].copy()
for a in range(n_faltantes):
    db2[falt[0,a]]=np.nan

print('valores faltantes',db2.isna().sum())
```

```
valores faltantes 1323
```

[42]:
```
db3=db2.copy()
dbrand=i_al(db3)
print(dbrand.isna().sum())
sns.kdeplot(db2)
sns.kdeplot(dbrand)
```

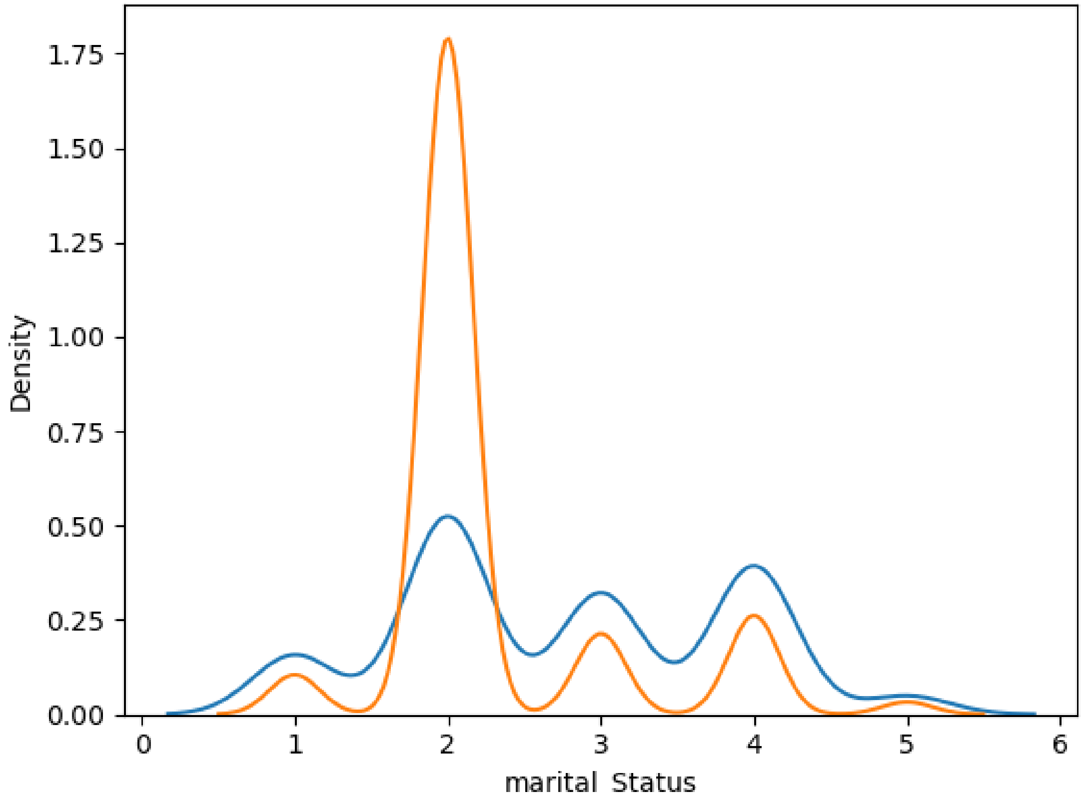
```
0
```

[42]: <AxesSubplot:xlabel='marital_Status', ylabel='Density'>



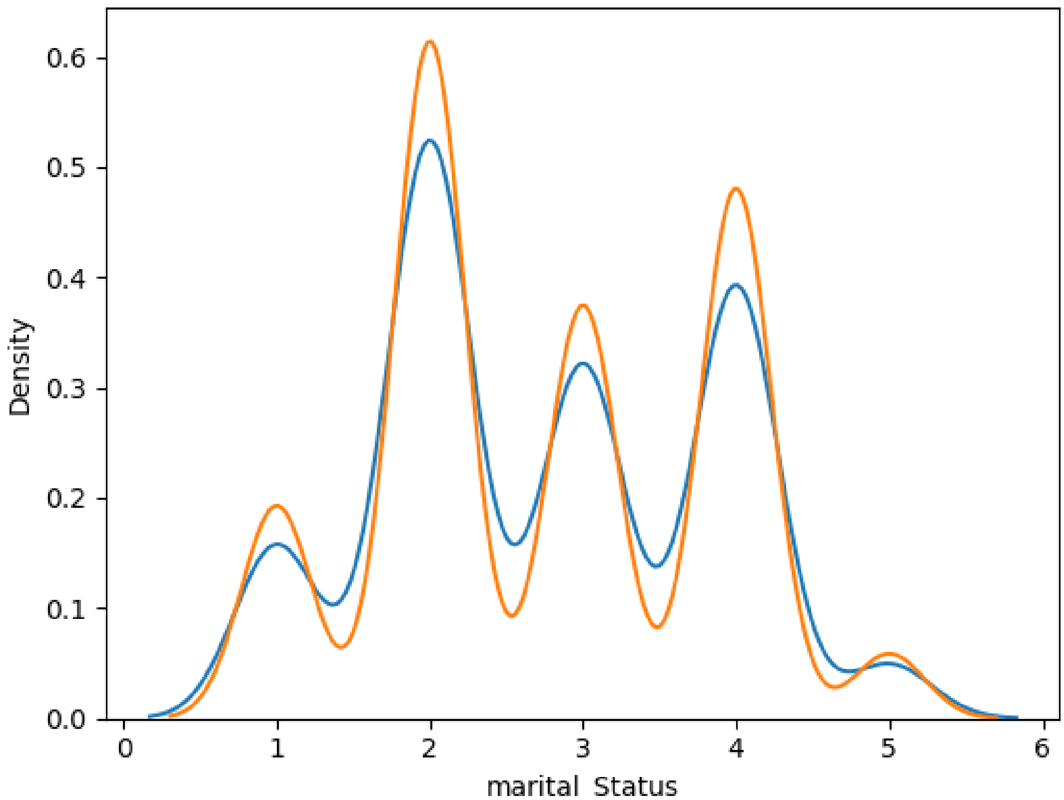
[43]:
```
db4=db2.copy()
dbmoda=moda(db4)
sns.kdeplot(db2)
sns.kdeplot(dbmoda)
print(dbmoda.isna().sum())
```

```
0
```



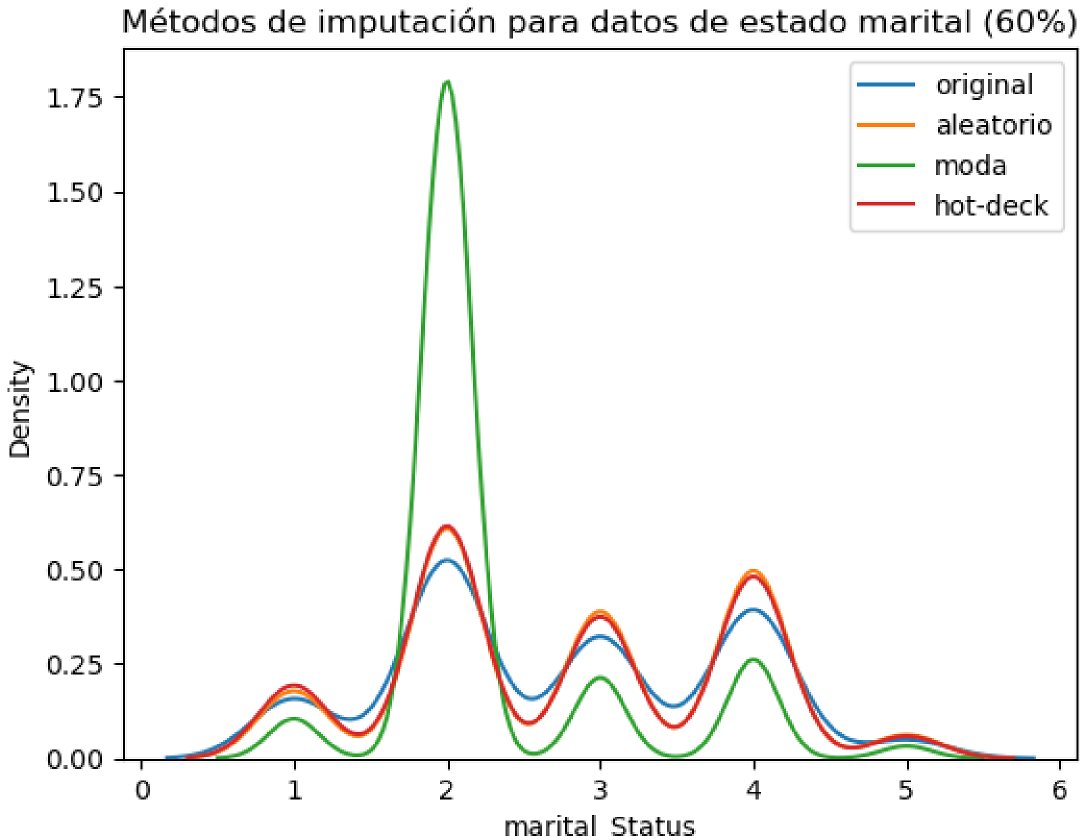
[44]:
```
db5=db2.copy()
dbhd=hot_deck(db5)
sns.kdeplot(db2)
sns.kdeplot(dbhd)
print(dbhd.isna().sum())
```

```
0
```



[45]:
```
## Distribuciones
sns.kdeplot(db2).set(title='Métodos de imputación para datos de estado marital␣
 ↪(60%)')
sns.kdeplot(dbrand)
sns.kdeplot(dbmoda)
sns.kdeplot(dbhd)

plt.legend(['original','aleatorio','moda','hot-deck'])
```

[45]:
```
<matplotlib.legend.Legend at 0x1e281d5d1c0>
```



2.1 Métricas de evaluación para datos continuos

Para las funciones continuas se tienen las siguientes métricas de evaluación

2.2 Error relativo porcentual

`$$e_{rp}=\frac{|\hat{y}-y|}{|\hat{y}|}*100\% $$`

2.3 RMSE

Error medio cuadrático. definido como:

`$$RMSE=\sqrt{\frac{\sum_{i=1}^n (\hat{y}_i-y_i)^2}{n}} $$`

donde:

- $\hat{y}_i$ es el valor verdadero de la serie de tiempo
- `$y_i $` es el valor imputado

2.4 MAE

error medio absoluto

$$MAE=\frac{1}{n}\sum_{i=1}^n |\hat{y}_i-y_i| $$

2.5 MAD

Desviación absoluto de la media

$$MAD=\frac{\sum_{i=1}^n |y_i - \bar{y}|}{n} $$

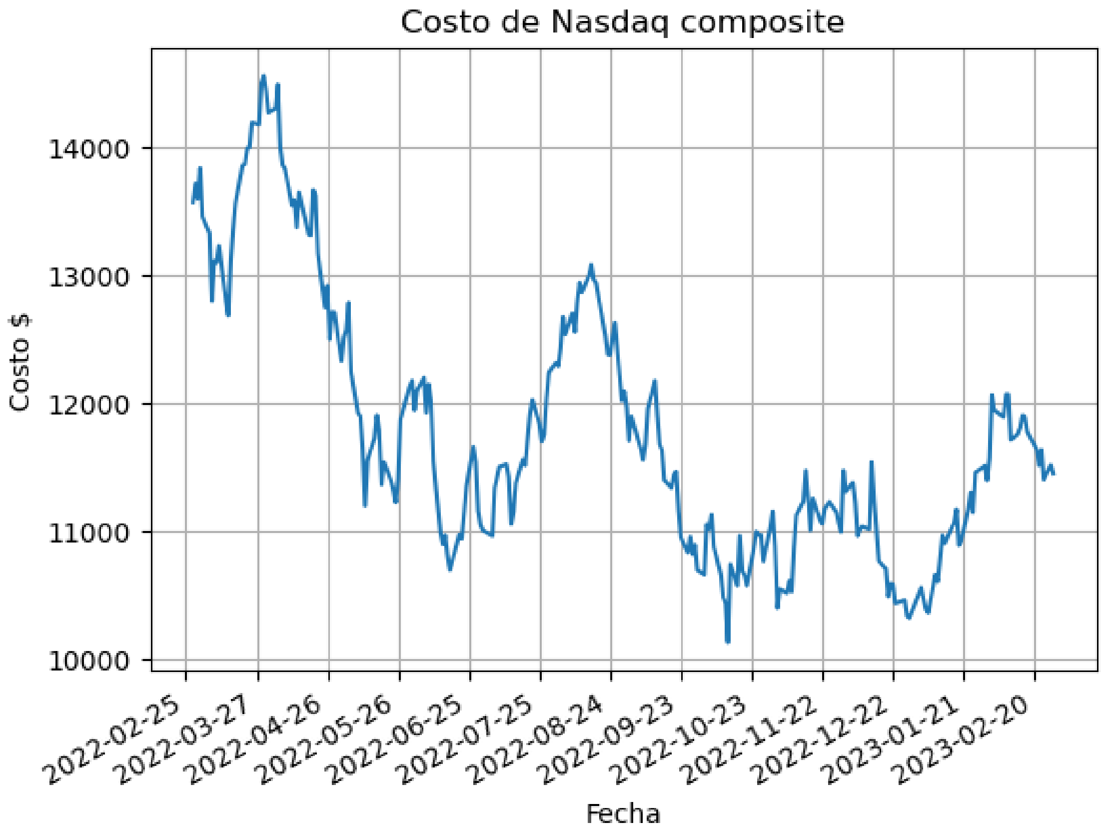
[46]:
```
def e_rp(yh,yi):
    return (abs(yh-yi))/abs(yh)

def rmse(yh,yi):
    n=len(yi)
    temp=0
    for a in range(n):
        temp=temp+(yh[a]-yi[a])**2
    res=np.sqrt(temp/n)
    return res

def mae(yh,yi):
    temp=0
    for a in range(len(yh)):
        temp=temp+abs(yh[a]-yi[a])
    return temp/len(yh)

def mad(yi):
    yb=sum(yi)/len(yi)
    temp=0
    for a in range(len(yi)):
        temp=temp+abs(yi[a]-yb)
    return temp/len(yi)
```

3 Imputación de datos

3.1 Continuos

[47]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import datetime as dt
## Importanto las bases de datos
cont_data=pd.read_csv('C:\\Users\\aldoa\\Machine Learning\\Practica 2 Imputación␣
↪de datos\\Nasdaq.csv')
#https://finance.yahoo.com/quote/%5EIXIC/history?p=%5EIXIC
#cont_data=pd.DataFrame(cont_data).set_index('Index')
cont_data=cont_data.to_numpy()
```

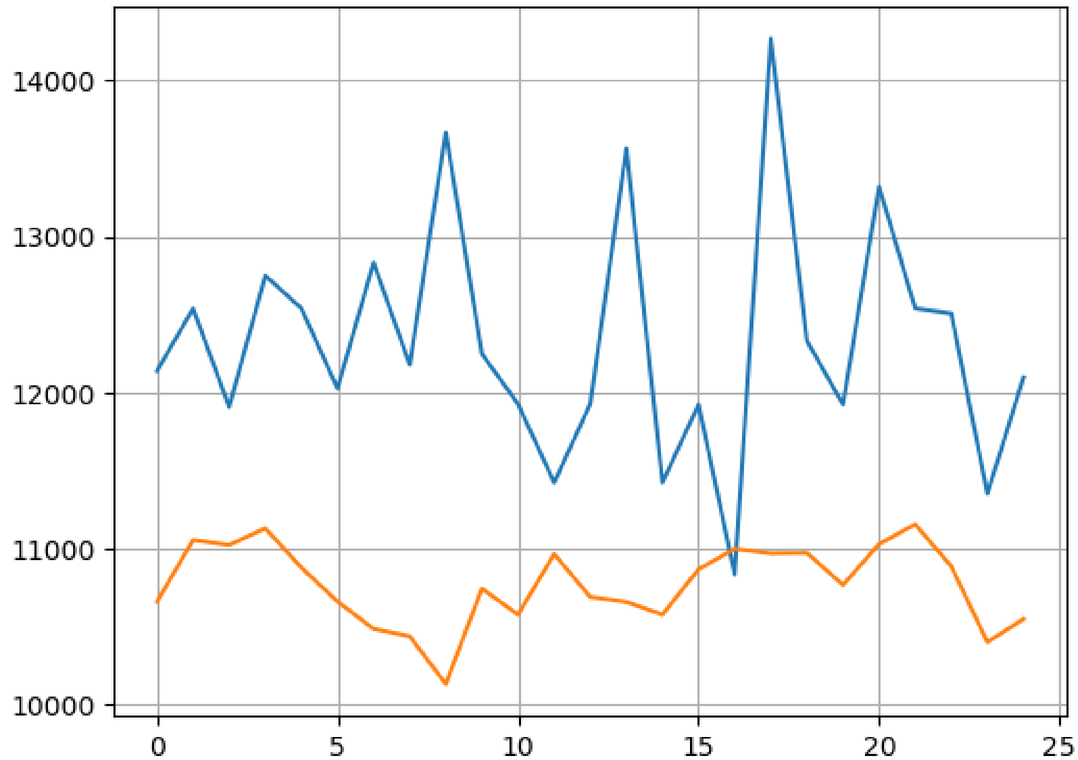
[48]:
```
x = [dt.strptime(d,'%Y-%m-%d').date() for d in cont_data[:,0]]
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=30))
plt.plot(x,cont_data[:,1])
plt.gcf().autofmt_xdate()
plt.xlabel('Fecha')
plt.ylabel('Costo $')
plt.title('Costo de Nasdaq composite')
plt.grid()
plt.show()
```



3.2 Método aleatorio

[49]:
```
import random

def aleatoria(val,pred):
    for a in range(pred):
        val.append(random.choice(val))
    return val
```

```
cd=(cont_data[:,1]-np.amin(cont_data[:,1]))/(np.amax(cont_data[:,1])-np.
 ↪amin(cont_data[:,1]))
precios=cont_data[:,1].tolist()

p_imputado_aleatorio=aleatoria(precios[:150],25)
plt.plot(p_imputado_aleatorio[150:175])
plt.plot(precios[150:175])
plt.grid()
```



3.3 Método de cadenas de Markov de primer orden y de corto plazo

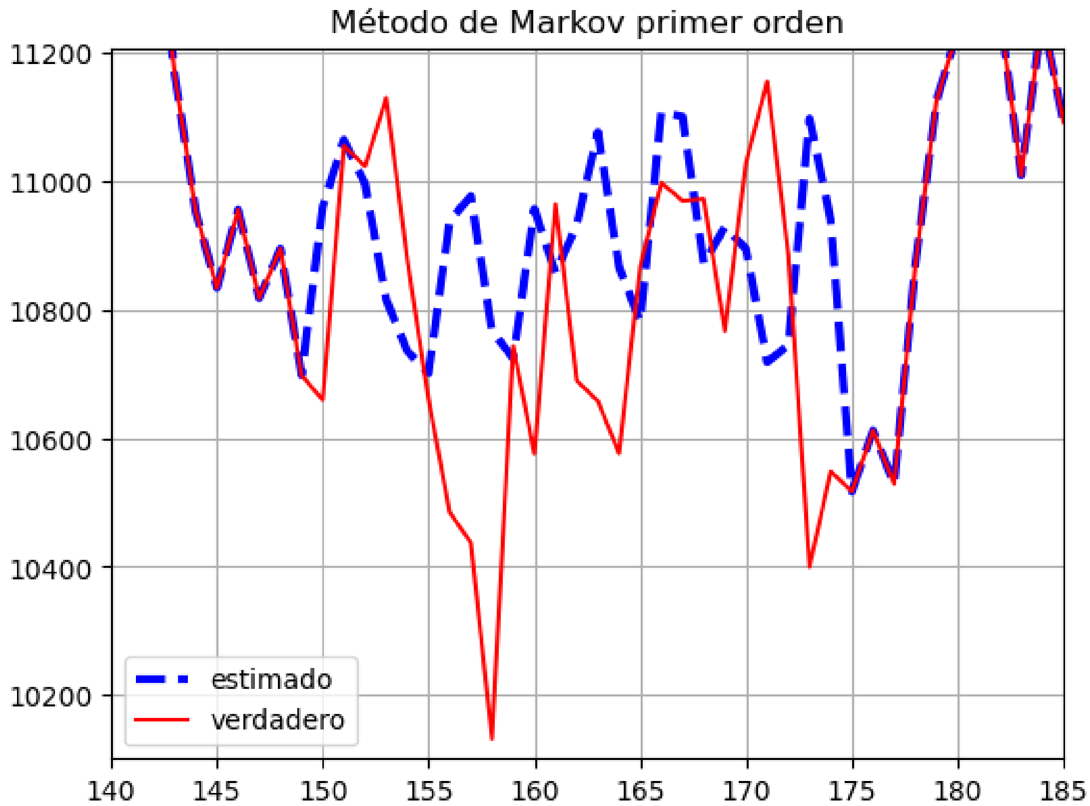
Para realizar este método primero se debe realizar una categorización en subintervalos entre el valor máximo y el minimo.

1. Posteriormente se realiza la matriz de transición m_t (codificación de datos a discretos).
2. Luego se toma en la matriz de estados s_t y se coloca la máxima probabilidad (1) donde se encuentre el último dato.
3. Se realiza la multiplicación de matrices $s_t \times m_t$.
4. Posteriormente se puede elegir un método de decodificación para obtener el valor requerido.
5. se actualiza la matriz de transición y se repite el paso 3

[50]:
```
def markov1_o1(val,bins,pred):
    bin_lims=np.linspace(min(val),max(val),bins)
    print(bin_lims)
    bin_indices=np.digitize(val,bin_lims)-1
    #print(val)
    #print(bin_indices)
    ## crear matriz de transición
    m_t=np.zeros((bins,bins))
    s_t=np.zeros((1,bins))
    s_t[0,bin_indices[-1]]=1
    for a in range(pred):
        m_t=np.zeros((bins,bins))
        for i in range(len(val)-1):
            ct=bin_indices[i]
            nt1=bin_indices[i+1]
            m_t[ct,nt1]+=1
        m_t=m_t/m_t.sum(axis=1,keepdims=True)
        #print(m_t)
        s_t=s_t.dot(m_t)
        ## Con el valor aleatorio maximo
        #print(s_t.dot(m_t))
        p_max=np.where(s_t==np.amax(s_t))
        #print(p_max)
        p_maxx=int(p_max[1])
        bin_indices=np.hstack((bin_indices,p_maxx))
        if p_maxx==bin_lims[-1]:
            val.append(random.
↪uniform(bin_lims[p_maxx],bin_lims[p_maxx]+(bin_lims[p_maxx]-bin_lims[p_maxx-1])))
        else:
            val.append(random.uniform(bin_lims[p_maxx],bin_lims[p_maxx+1]))

    return val
```

[51]:
```
inicio=150
n_p=25
pred=markov1_o1(precios[:inicio],10,n_p) ## este es el bueno
pred2=pred+precios[inicio+n_p:]
plt.plot(pred2,'b--',linewidth=3)
plt.plot(precios,'r')
plt.xlim([inicio-10,inicio+n_p+10])
plt.ylim([min(pred[inicio:inicio+n_p])-600,max(pred[inicio:inicio+n_p])+100])
plt.legend(['estimado','verdadero'])
plt.title('Método de Markov primer orden')
plt.grid()
#result= np.where((markov1_o1(precios,10,1)==np.amax(markov1_o1(precios,10,1))))
#print(result)
#print(int(result[1]))
```

```
[10697.549805   11126.55425378 11555.55870256 11984.56315133
 12413.56760011 12842.57204889 13271.57649767 13700.58094644
 14129.58539522 14558.589844   ]
```



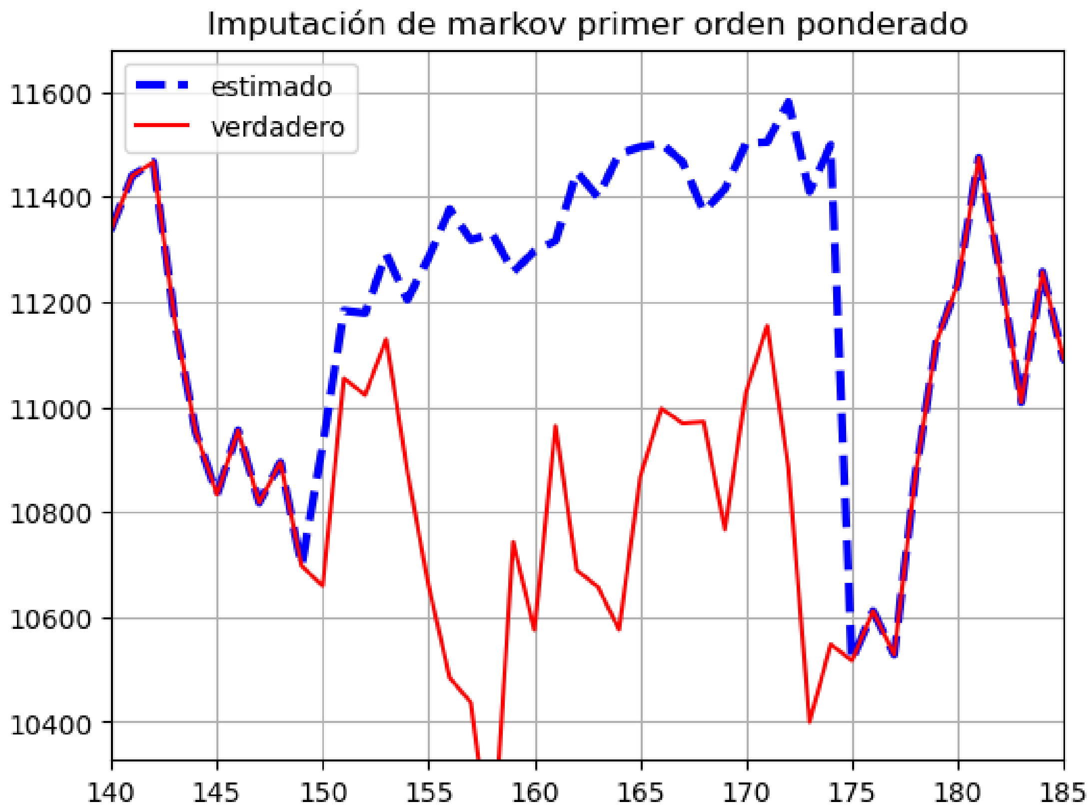
[52]:
```python
## markov con
def markov2_o1(val,bins,pred):
    bin_lims=np.linspace(min(val),max(val),bins)
    print(bin_lims)
    bin_indices=np.digitize(val,bin_lims)-1

    #print(len(bin_indices))
    ## crear matriz de transición
    m_t=np.zeros((bins,bins))
    s_t=np.zeros((1,bins))
    s_t[0,bin_indices[-1]]=1
    for a in range(pred):
        m_t=np.zeros((bins,bins))
        for i in range(len(val)-1):
            ct=bin_indices[i]
            nt1=bin_indices[i+1]
            m_t[ct,nt1]+=1
```

```
        m_t=m_t/m_t.sum(axis=1,keepdims=True)

        s_t=s_t.dot(m_t)
        p_max=np.where(s_t==np.amax(s_t))
        #print(p_max)
        p_maxx=int(p_max[1])
        bin_indices=np.hstack((bin_indices,p_maxx))
        ## Con pesos del valor del
        temp=0

        for tt in range(bins):
            if tt+1==bins:
                temp=temp+(s_t[0,tt]*random.
↪uniform(bin_lims[tt-1],bin_lims[tt]+(bin_lims[tt-1]-bin_lims[tt-2])))
            else:
                temp=temp+(s_t[0,tt]*random.uniform(bin_lims[tt],bin_lims[tt+1]))
        val.append(temp)

    return val
```

[53]:

```
inicio=150
n_p=25
pred12=markov2_o1(precios[:inicio],10,n_p)
pred2=pred12+precios[inicio+n_p:]
plt.plot(pred2,'b--',linewidth=3)
plt.plot(precios,'r')
plt.xlim([inicio-10,inicio+n_p+10])
plt.ylim([min(pred12[inicio:inicio+n_p])-600,max(pred12[inicio:inicio+n_p])+100])
plt.legend(['estimado','verdadero'])
plt.title('Imputación de markov primer orden ponderado')
plt.grid()
```

```
[10697.549805   11126.55425378 11555.55870256 11984.56315133
 12413.56760011 12842.57204889 13271.57649767 13700.58094644
 14129.58539522 14558.589844  ]
```



3.4 Markov de segundo orden

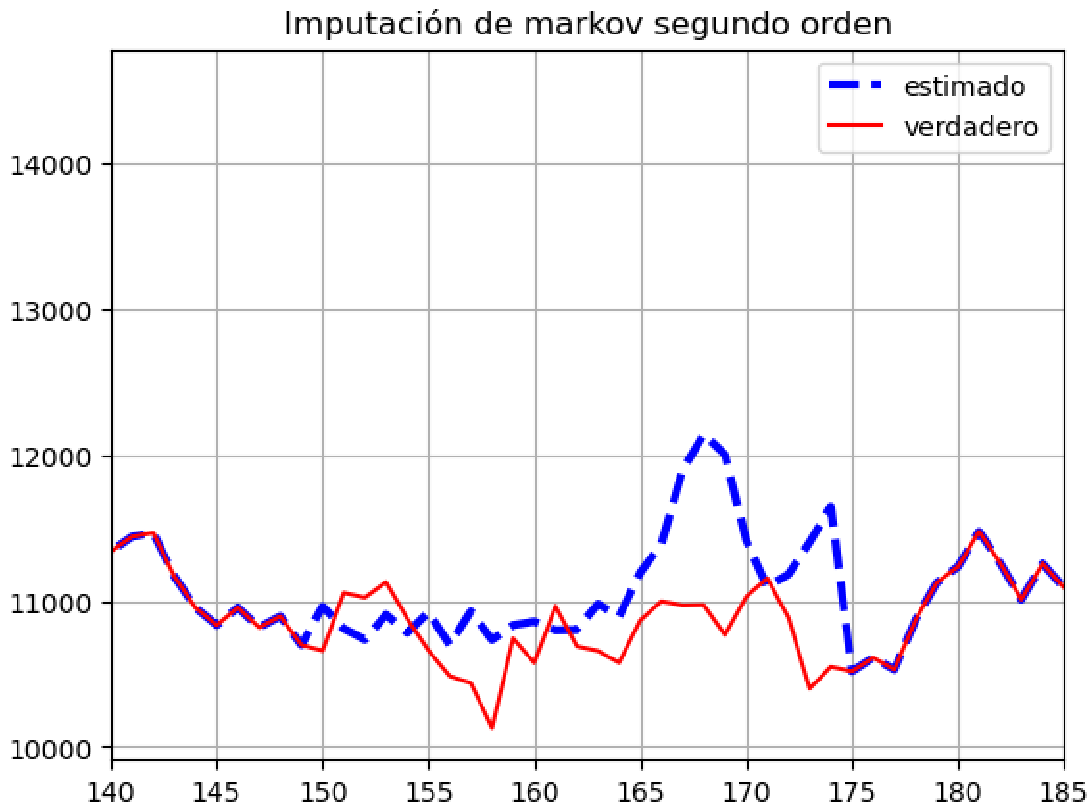
se agrega un eje extra para conocer el valor anterior se obtiene a partir del cruce de probabilidades el nuevo estado.

[54]:
```
def markov_o2(val,bins,pred):
    bin_lims=np.linspace(min(val),max(val),bins)
    #print(bin_lims)
    bin_indices=np.digitize(val,bin_lims)-1

    #print(len(bin_indices))
    ## crear matriz de transición
    m_t=np.zeros((bins,bins,bins))
    s_t=np.zeros((1,bins**2))
    s_t[0,bin_indices[-1]]=0.9
    s_t[0,bin_indices[-2]]=0.1
    lbins=[]
    for r in range(bins):
        lbins.append(r)
    for a in range(pred):
        m_t=np.zeros((bins,bins,bins))
```

```
        for i in range(2,len(val)): ## Agregar la siguiente parte
            ct=bin_indices[i-2] # estado anterior
            nt1=bin_indices[i-1] # estado actual
            nt2=bin_indices[i] # estado siguiente

            m_t[ct,nt1,nt2]+=1

        # Obtener las sumas de cada fila sin contar los ceros
        row_sums = np.sum(m_t, axis=2)
        nonzero_rows = np.count_nonzero(m_t, axis=2)
        row_sums[nonzero_rows == 0] = 1 # Asignar 1 a las filas que solo␣
↪contienen ceros

        # Dividir cada fila por su suma
        m_t = m_t / row_sums[:,:,np.newaxis]
        #print(m_t.shape)
        #m_t=m_t/m_t.sum(axis=2,keepdims=True)
        ## predicciones
        e_2=bin_indices[-2]
        e_1=bin_indices[-1]
        probs=m_t[e_2,e_1,:]
        next_state=np.random.choice(lbins,p=probs)
        bin_indices=np.hstack((bin_indices,next_state))
        p_maxx=next_state
        if p_maxx==bin_lims[-1]:
            val.append(random.
↪uniform(bin_lims[p_maxx],bin_lims[p_maxx]+(bin_lims[p_maxx]-bin_lims[p_maxx-1])))
        else:
            val.append(random.uniform(bin_lims[p_maxx],bin_lims[p_maxx+1]))
        """
        p_maxx=int(p_max[1])
        bin_indices=np.hstack((bin_indices,p_maxx))
        ## Con pesos del valor del
        temp=0

        for tt in range(bins):
            if tt+1==bins:
                temp=temp+(s_t[0,tt]*random.
↪uniform(bin_lims[tt-1],bin_lims[tt]+(bin_lims[tt-1]-bin_lims[tt-2])))
            else:
                temp=temp+(s_t[0,tt]*random.uniform(bin_lims[tt],bin_lims[tt+1]))
        val.append(temp)
        """
    return val
```

[55]:
```
inicio=150
n_p=25
```

```
pred_m2=markov_o2(precios[:inicio],28,n_p)
pred2=pred_m2+precios[inicio+n_p:]
plt.plot(pred2,'b--',linewidth=3)
plt.plot(precios,'r')
plt.xlim([inicio-10,inicio+n_p+10])
#plt.ylim([min(pred[inicio:inicio+n_p])-600,max(pred[inicio:inicio+n_p])+100])
plt.legend(['estimado','verdadero'])
plt.title('Imputación de markov segundo orden')
plt.grid()
```

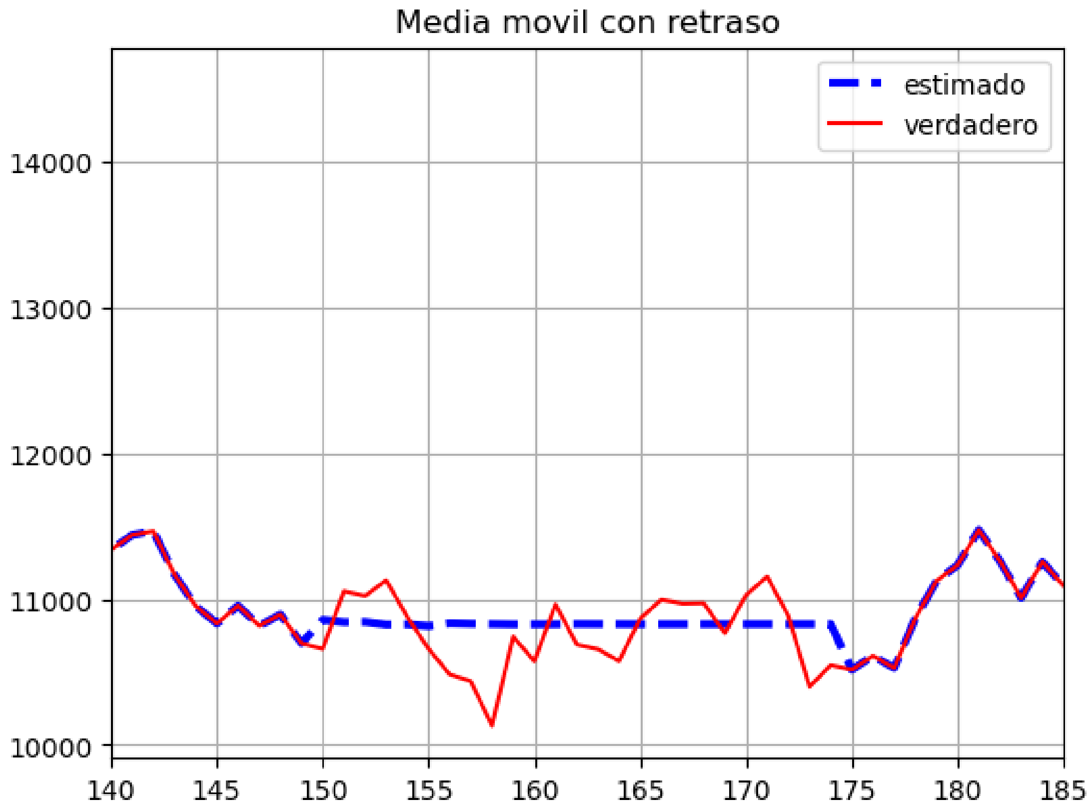


3.5 Media móvil con retraso y ponderada

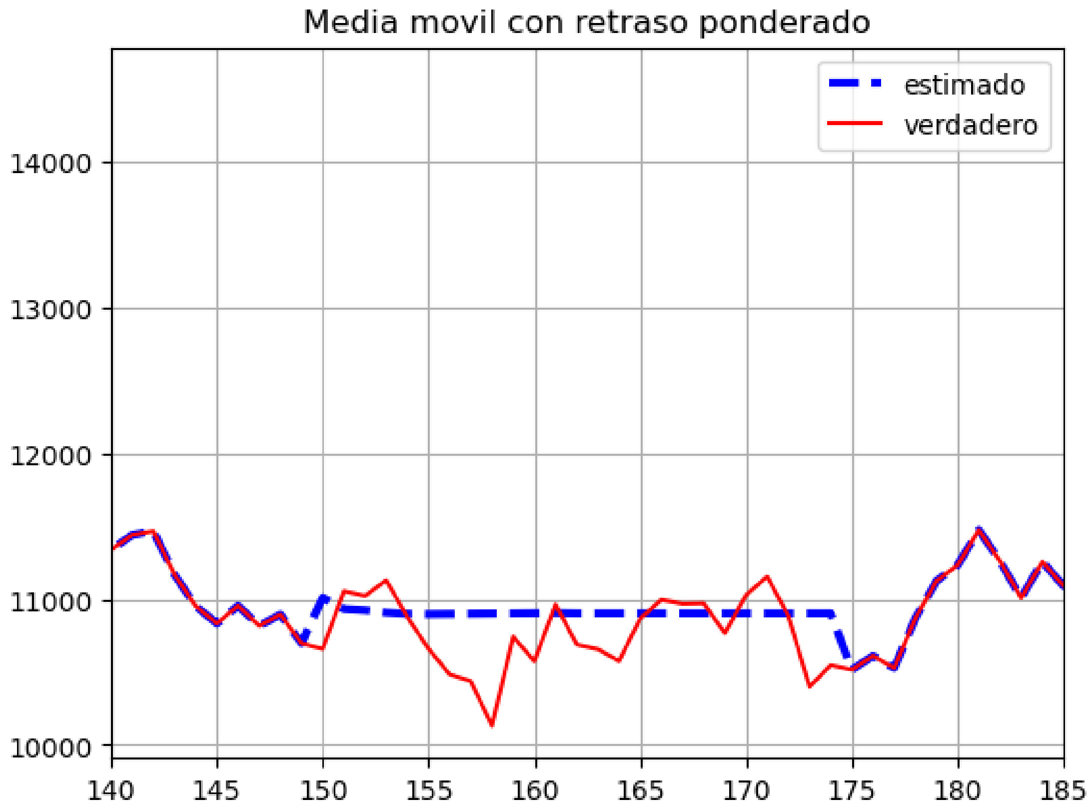
[56]:
```
def m_movil(retraso,val,n_pred):
    temp=0

    for a in range(n_pred):
        l=len(val)-1
        temp=0
        for b in range(retraso):
            temp=temp+val[l-b]
```

```
        temp/=retraso
        val.append(temp)
    return val


def m_movil_p(retraso,val,n_pred):
    temp=0
    j=(retraso*(retraso+1))/2
    for a in range(n_pred):
        l=len(val)-1
        temp=0
        for b in range(retraso):
            temp=temp+(((b+1)*val[l-retraso+b])/j)
        val.append(temp)
    return val
```

[57]:
```
inicio=150
n_p=25
pred_m_movil=m_movil(6,precios[:inicio],n_p) ##
pred2_m=pred_m_movil+precios[inicio+n_p:]
plt.plot(pred2_m,'b--',linewidth=3)
plt.plot(precios,'r')
plt.xlim([inicio-10,inicio+n_p+10])
#plt.ylim([min(pred[inicio:inicio+n_p])-600,max(pred[inicio:inicio+n_p])+100])
plt.legend(['estimado','verdadero'])
plt.title('Media movil con retraso')
plt.grid()
```

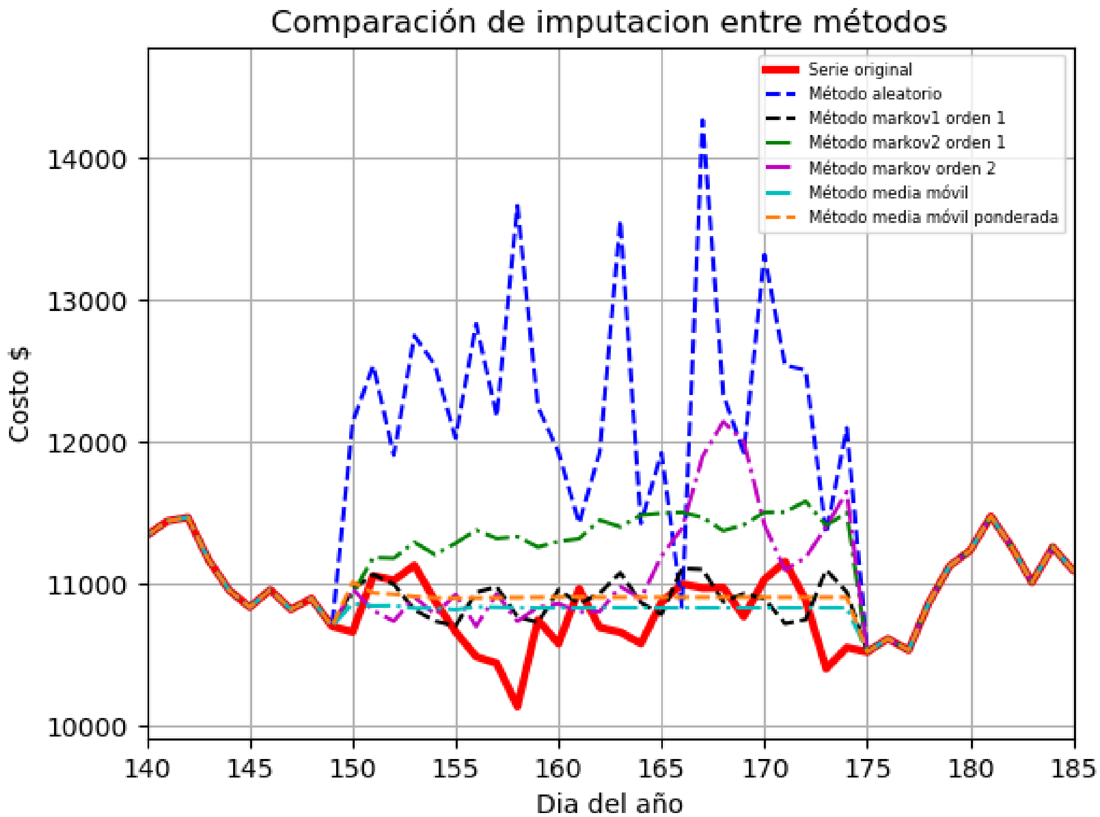


[58]:
```
## Media movil con retraso ponderado
inicio=150
n_p=25
pred_m_movil_p=m_movil_p(10,precios[:inicio],n_p) ##
pred2_mp=pred_m_movil_p+precios[inicio+n_p:]
plt.plot(pred2_mp,'b--',linewidth=3)
plt.plot(precios,'r')
plt.xlim([inicio-10,inicio+n_p+10])
#plt.ylim([min(pred[inicio:inicio+n_p])-600,max(pred[inicio:inicio+n_p])+100])
plt.legend(['estimado','verdadero'])
plt.title('Media movil con retraso ponderado')
plt.grid()
```



[62]:
```
## inicio 150 con 25 predicciones y fin en 175
p_imputado_aleatorio # aleatorio
pred # markov1 orden 1
pred12 # markov2 orden 1
pred_m2 # markov orden 2
pred_m_movil # media movil
pred_m_movil_p # media movil ponderada

# Gráficas

plt.plot(precios,'r',linewidth=3)
plt.plot(p_imputado_aleatorio+precios[inicio+n_p:],'b--')
plt.plot(pred+precios[inicio+n_p:],'k--')
plt.plot(pred12+precios[inicio+n_p:],'g-.')
plt.plot(pred_m2+precios[inicio+n_p:],'m-.')
plt.plot(pred_m_movil+precios[inicio+n_p:],'c-.')
plt.plot(pred_m_movil_p+precios[inicio+n_p:],'tab:orange',ls='--')
plt.xlim([inicio-10,inicio+n_p+10])
plt.legend(['Serie original','Método aleatorio','Método markov1 orden 1',
            'Método markov2 orden 1','Método markov orden 2','Método media␣
 ↪móvil',
```

```
             'Método media móvil ponderada'],prop={'size':6},loc='upper right')
plt.xlabel('Dia del año')
plt.ylabel('Costo $')
plt.title('Comparación de imputacion entre métodos')
plt.grid()
```



[64]:
```
## Métricas para series de puntos
## rmse
rmc=[]
a1=rmse(precios[inicio:inicio+n_p],p_imputado_aleatorio[inicio:inicio+n_p])
rmc.append(a1)
a1=rmse(precios[inicio:inicio+n_p],pred[inicio:inicio+n_p])
rmc.append(a1)
a1=rmse(precios[inicio:inicio+n_p],pred12[inicio:inicio+n_p])
rmc.append(a1)
a1=rmse(precios[inicio:inicio+n_p],pred_m2[inicio:inicio+n_p])
rmc.append(a1)
a1=rmse(precios[inicio:inicio+n_p],pred_m_movil[inicio:inicio+n_p])
rmc.append(a1)
a1=rmse(precios[inicio:inicio+n_p],pred_m_movil_p[inicio:inicio+n_p])
```

```
rmc.append(a1)

# MAE
maec=[]
a1=mae(precios[inicio:inicio+n_p],p_imputado_aleatorio[inicio:inicio+n_p])
maec.append(a1)
a1=mae(precios[inicio:inicio+n_p],pred[inicio:inicio+n_p])
maec.append(a1)
a1=mae(precios[inicio:inicio+n_p],pred12[inicio:inicio+n_p])
maec.append(a1)
a1=mae(precios[inicio:inicio+n_p],pred_m2[inicio:inicio+n_p])
maec.append(a1)
a1=mae(precios[inicio:inicio+n_p],pred_m_movil[inicio:inicio+n_p])
maec.append(a1)
a1=mae(precios[inicio:inicio+n_p],pred_m_movil_p[inicio:inicio+n_p])
maec.append(a1)
#MAD
madc=[]
a1=mad(p_imputado_aleatorio[inicio:inicio+n_p])
madc.append(a1)
a1=mad(pred[inicio:inicio+n_p])
madc.append(a1)
a1=mad(pred12[inicio:inicio+n_p])
madc.append(a1)
a1=mad(pred_m2[inicio:inicio+n_p])
madc.append(a1)
a1=mad(pred_m_movil[inicio:inicio+n_p])
madc.append(a1)
a1=mad(pred_m_movil_p[inicio:inicio+n_p])
madc.append(a1)
print('rmse: ',rmc)
print('mae: ',maec)
print('mad: ',madc)
print('mad real:',mad(precios[inicio:inicio+n_p]))
```

```
rmse:  [1752.2246327376595, 317.9567113716203, 655.1019263044956,
559.1797793548402, 260.1307804299107, 288.34349054782945]
mae:  [1570.0339062000005, 252.36607860447447, 591.5113166050114,
436.53238390646214, 217.054585838251, 227.49350757841]
mad:  [580.7418626208004, 107.69166741924339, 111.20804569041233,
335.1903624807304, 4.624181215804128, 11.305144053204423]
mad real: 214.5460594464001
```

[61]:
```
## Medición por puntos
#datos=cont_data[:,1].tolist()
rnd=np.random.randint(100,len(precios)-1,(5,1))
rnd2=rnd.tolist()
```

```
res_al=[]
res_mk11=[]
res_mk21=[]
res_mk12=[]
res_mm=[]
res_mmp=[]
er=[]
resv=[]
print(rnd)
print('valor verdadero')
for u in range(len(rnd2)):
    resv.append(precios[rnd[u,0]])
print(resv)
print("*******")
for t in range(len(rnd2)):
    r_al=aleatoria(precios[:rnd[t,0]],1)
    res_al.append(r_al[-1])

    r_mar11=markov1_o1(precios[:rnd[t,0]],10,1)
    res_mk11.append(r_mar11[-1])

    r_mar21=markov2_o1(precios[:rnd[t,0]],10,1)
    res_mk21.append(r_mar21[-1])

    r_mar12=markov_o2(precios[:rnd[t,0]],10,1)
    res_mk12.append(r_mar12[-1])

    r_mm=m_movil(6,precios[:rnd[t,0]],1)
    res_mm.append(r_mm[-1])

    r_mmp=m_movil_p(6,precios[:rnd[t,0]],1)
    res_mmp.append(r_mmp[-1])
print("*** Resultados ****")
print(res_al)
print(res_mk11)
print(res_mk21)
print(res_mk12)
print(res_mm)
print(res_mmp)
```

```
[[126]
 [202]
 [236]
 [183]
 [237]]
valor verdadero
[12021.049805, 11012.620117, 11904.410156, 11008.669922, 11891.25]
*******
```

```
[10697.549805   11126.55425378 11555.55870256 11984.56315133
 12413.56760011 12842.57204889 13271.57649767 13700.58094644
 14129.58539522 14558.589844  ]
[10697.549805   11126.55425378 11555.55870256 11984.56315133
 12413.56760011 12842.57204889 13271.57649767 13700.58094644
 14129.58539522 14558.589844  ]
[10131.820313   10623.68359422 11115.54687544 11607.41015667
 12099.27343789 12591.13671911 13083.00000033 13574.86328156
 14066.72656278 14558.589844  ]
[10131.820313   10623.68359422 11115.54687544 11607.41015667
 12099.27343789 12591.13671911 13083.00000033 13574.86328156
 14066.72656278 14558.589844  ]
[10131.820313   10623.68359422 11115.54687544 11607.41015667
 12099.27343789 12591.13671911 13083.00000033 13574.86328156
 14066.72656278 14558.589844  ]
[10131.820313   10623.68359422 11115.54687544 11607.41015667
 12099.27343789 12591.13671911 13083.00000033 13574.86328156
 14066.72656278 14558.589844  ]
[10131.820313   10623.68359422 11115.54687544 11607.41015667
 12099.27343789 12591.13671911 13083.00000033 13574.86328156
 14066.72656278 14558.589844  ]
[10131.820313   10623.68359422 11115.54687544 11607.41015667
 12099.27343789 12591.13671911 13083.00000033 13574.86328156
 14066.72656278 14558.589844  ]
[10131.820313   10623.68359422 11115.54687544 11607.41015667
 12099.27343789 12591.13671911 13083.00000033 13574.86328156
 14066.72656278 14558.589844  ]
[10131.820313   10623.68359422 11115.54687544 11607.41015667
 12099.27343789 12591.13671911 13083.00000033 13574.86328156
 14066.72656278 14558.589844  ]
*** Resultados ****
[12239.69043, 11351.30957, 11398.580078, 13716.700195, 10869.169922]
[12678.867921141586, 11369.932257682754, 11657.747313255184, 11246.389645847143,
12006.501974105417]
[12551.07770360393, 11242.053446347738, 11729.581337429048, 11322.956261393008,
11735.39305681399]
[12979.339620146213, 10849.167997297209, 12041.0885160921, 11359.409199560996,
11238.060761741033]
[12541.316731666666, 11136.643392, 11661.090006666667, 11081.865071666667,
11733.4134115]
[12505.107747333333, 11175.11579252381, 11656.53715595238, 11133.017345809523,
11761.445824190474]
```

[ ]:

```

```