Resumen del capítulo: Pronóstico de series temporales

Pronóstico de series temporales

El objetivo del **pronóstico de series temporales** es desarrollar un modelo que prediga los valores futuros de una serie temporal con base en datos anteriores. El periodo en el futuro para el que se prepara el pronóstico se conoce como **horizonte de pronóstico**. Para los ejercicios de este capítulo usaremos un horizonte de un paso.

Si los valores de una serie temporal o de la función `x(t)` (donde t = tiempo) son números, entonces te enfrentas a una tarea de regresión para la serie temporal. Si son categorías, será una tarea de clasificación.

Crearemos un conjunto de entrenamiento y un conjunto de prueba usando los datos iniciales. No puedes mezclar los conjuntos en el pronóstico de series temporales, y los datos del conjunto de entrenamiento deben preceder a los datos del conjunto de prueba. De lo contrario, las pruebas del modelo serán defectuosas: después de todo, el modelo no debe entrenarse con datos futuros. La función `train_test_split()` del módulo `sklearn.model_selection` mezcla datos de forma predeterminada, así que vamos a establecer el argumento `shuffle` en `False` para que los datos se puedan separar correctamente en conjuntos de entrenamiento y de prueba:

```
import pandas as pd
from sklearn.model_selection import train_test_split

train, test = train_test_split(data, shuffle=False, test_size=0.2)
```

Exactitud del pronóstico

Usaremos la métrica *EAM* para evaluar los modelos en nuestras tareas, ya que esta métrica es fácil de interpretar.

Hay dos formas de pronosticar series temporales sin entrenamiento:

1. Todos los valores de la muestra de prueba se pronostican con el mismo número (una constante). Para la métrica *EAM*, este número es la mediana.
2. El nuevo valor *x(t)* se predice mediante el valor anterior de la serie, definido como *x(t-1)*. Este método es independiente de la métrica.

Creación de características

1. Características del calendario

Por lo general, las tendencias y la estacionalidad se vinculan con una fecha específica. El tipo `datetime64` en pandas ya contiene la información necesaria y todo lo que resta es presentarlo como columnas separadas. Veamos este ejemplo:

```
# esta característica contiene años como valores numéricos
data['year'] = data.index.year

# esta característica contiene días de la semana como valores numéricos
data['dayofweek'] = data.index.dayofweek
```

2. Características de desfase

Los valores anteriores en la serie temporal te dirán si la función `x(t)` aumentará o disminuirá. Vamos a usar la función `shift()` para obtener los valores de desfase:

```
data['lag_1'] = data['target'].shift(1)
data['lag_2'] = data['target'].shift(2)
data['lag_3'] = data['target'].shift(3)
```

No todos los valores de desfase están disponibles para las primeras fechas, por lo que estas líneas contienen `NaN`.

3. Media móvil

La característica de media móvil establece la tendencia general de la serie temporal. Aquí podrás recordar cómo calcularla:

```
data['rolling_mean'] = data['target'].rolling(5).mean()
```

La media móvil en *t* considera el valor actual de la serie *x(t)*. Esto es incorrecto: el objetivo se "deslizó" en las características. El cálculo de la media móvil no debe incluir el valor actual de la serie.