01Introduction_Tutorial

February 11, 2022

1 QBUS3850 Lab 1 Tasks

This tutorial will cover reading in data using pandas, understanding how dates and times work in Python and implementing an expanding window. The dataset that we will use is the electricity dataset from lectures (this can be found on Canvas). We will evaluate forecasts from simple exponential smoothing, Holt's method and the Holt Winters' additive method.

1.1 Data and data types

1. Read the data from the file into a variable called `df`.
2. Check the type of each variable by running `df.dtypes`. Is the variable `SETTLEMENTDATE` a datetime or an object?
3. If it is an object, convert it to a datetime.

The function `read_csv()` can be used to read in the data, make sure that the file electricity.csv is in your working directory otherwise provide a full path to the file.

[1]:
```
import pandas as pd
df = pd.read_csv('electricity.csv')
df.dtypes
```

[1]:
```
REGION              object
SETTLEMENTDATE      object
TOTALDEMAND        float64
RRP                float64
PERIODTYPE          object
dtype: object
```

By default, `read_csv()` has read in `SETTLEMENTDATE` as the same type as `REGION`, i.e. as a string and not a datetime. Sometimes we can add the argument `parse_dates=True` but that will not work in this case. Instead to coerce `SETTLEMENTDATE` to a datetime run the following:

[2]:
```
df['SETTLEMENTDATE'] = pd.to_datetime(df['SETTLEMENTDATE'])
df.dtypes
```

[2]:
```
REGION                      object
SETTLEMENTDATE      datetime64[ns]
TOTALDEMAND                float64
RRP                        float64
```

```
PERIODTYPE                  object
dtype: object
```

Note now that the data type of SETTLEMENTDATE is a datetime.

1.1.1 Forecast exercise

1. Using data from April 1, 00:30 to April 25, 00:00 as training data generate forecasts for the next 6 hours (12 half hour periods) using:

- Simple Exponential Smoothing
- Holt's Method
- Holt Winters' additive method

2. Compute the squared error (i.e. $(y_{t+h} - \hat{y}_{t+h})^2$) for each method at each horizon

[3]:
```python
#datetime needed to manipulate dates
import datetime
#numpy needed to work with vectors
import numpy as np
#statmodels needed for models
from statsmodels.tsa.holtwinters import SimpleExpSmoothing, Holt,␣
↪ExponentialSmoothing

#longest horizon in half hour steps
hmax = 12
#full horizon as a time interval
fullhor = datetime.timedelta(hours=hmax/2)
#construct end of training period as datetime
endtrain = datetime.datetime(2021,4,25,0,0)

#Filter training data
train = df[(df['SETTLEMENTDATE']<=endtrain)]
#Filter test data
test = df[(df['SETTLEMENTDATE']>endtrain) &␣
↪(df['SETTLEMENTDATE']<=(endtrain+fullhor))]

#Simple Exponential Smoothing
#Specify model
model = SimpleExpSmoothing(np.asarray(train['TOTALDEMAND']))
#Fit Model
fit_ses = model.fit()
#Make forecasts
fc_ses = fit_ses.forecast(hmax)
#Compute square error
sqerr_ses = np.square(fc_ses-np.asarray(test['TOTALDEMAND']))
```

```python
#Holt's Method (same steps as above)
model = Holt(np.asarray(train['TOTALDEMAND']))
fit_holt = model.fit()
fc_holt = fit_holt.forecast(hmax)
sqerr_holt = np.square(fc_holt-np.asarray(test['TOTALDEMAND']))

#Holt Winters' Method  (same steps as above)
model = ExponentialSmoothing(np.
 ↪asarray(train['TOTALDEMAND']),trend='add',seasonal='add',seasonal_periods=48)
fit_hw = model.fit()
fc_hw = fit_hw.forecast(hmax)
sqerr_hw = np.square(fc_hw-np.asarray(test['TOTALDEMAND']))

#Initialise Results data frame
res = pd.DataFrame()
res['h']=pd.Series(range(1,hmax+1))
res['SES']=pd.Series(sqerr_ses)
res['Holt']=pd.Series(sqerr_holt)
res['HW']=pd.Series(sqerr_hw)

print(res)
```

```
/home/anastasios/anaconda3/lib/python3.8/site-
packages/statsmodels/tsa/holtwinters/model.py:427: FutureWarning: After 0.13
initialization must be handled at model creation
  warnings.warn(
/home/anastasios/anaconda3/lib/python3.8/site-
packages/statsmodels/tsa/holtwinters/model.py:920: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
```

	h	SES	Holt	HW
0	1	3.610507e+04	3763.606543	264.570208
1	2	9.428790e+04	2530.230144	14649.836518
2	3	2.279140e+05	8564.493031	37767.913360
3	4	4.493736e+05	24774.035755	46989.478791
4	5	7.990632e+05	63933.553593	47745.020376
5	6	1.016857e+06	57237.694216	48849.337901
6	7	1.209457e+06	41009.149339	8757.172245
7	8	1.299197e+06	13105.741806	6598.925882
8	9	1.177645e+06	4657.589539	49315.704505
9	10	1.088774e+06	56688.458486	143792.551138
10	11	9.784837e+05	176778.394837	346867.561012
11	12	7.638818e+05	440533.630875	573459.603995

Some things to notice:

- Holt Winters has the lowest square error at one step ahead
- Holt has the lowest square error 12-steps ahead

- Simple exponential smoothing has the worst squared errors at all horizons
- None of this is meaningful since we are only looking at a single instance of forecasts.

3. Repeat the same exercise but for an expanding window that expands one half hour at a time. Do this over 192 windows (i.e. four days). This will take a minute or two to run.
4. Compute Root Mean Square Error (RMSE) given by $RMSE = \sqrt{\frac{1}{192}\sum(y_{t+h}-\hat{y}_{t+h})^2}$ over all windowsfort each forcasting horizon.
5. Which method is the best at a one-step ahead horizon?
6. Which method is the best at a twelve-step ahead horizon?

[ ]:
```
#Switch off warnings
import warnings
warnings.filterwarnings('ignore')

#set number of windows
n_wind=192
#define window increment
windowinc = datetime.timedelta(minutes=30)

#Create list of dates
datetime_list = [endtrain+i*windowinc for i in range(n_wind)]

#Initialise vectors to store root mean squared error
rmse_ses=np.zeros(hmax)
rmse_holt=np.zeros(hmax)
rmse_hw=np.zeros(hmax)

#Loop

for i in datetime_list:
    train = df[(df['SETTLEMENTDATE']<=i)]
    test = df[(df['SETTLEMENTDATE']>i) & (df['SETTLEMENTDATE']<=(i+fullhor))]

    #Simple Exponential Smoothing
    model = SimpleExpSmoothing(np.asarray(train['TOTALDEMAND']))
    fit_ses = model.fit()
    fc_ses = fit_ses.forecast(hmax)
    sqerr_ses = np.square(fc_ses-np.asarray(test['TOTALDEMAND']))
    rmse_ses += sqerr_ses

    #Simple Exponential Smoothing
    model = Holt(np.asarray(train['TOTALDEMAND']))
    fit_holt = model.fit()
    fc_holt = fit_holt.forecast(hmax)
    sqerr_holt = np.square(fc_holt-np.asarray(test['TOTALDEMAND']))
    rmse_holt += sqerr_holt

    #Holt-Winters Smoothing
```

```python
    model = ExponentialSmoothing(np.
↪asarray(train['TOTALDEMAND']),trend='add',seasonal='add',seasonal_periods=48)
    fit_hw = model.fit()
    fc_hw = fit_hw.forecast(hmax)
    sqerr_hw = np.square(fc_hw-np.asarray(test['TOTALDEMAND']))
    rmse_hw += sqerr_hw


rmse_ses = np.sqrt(rmse_ses/n_wind)
rmse_holt = np.sqrt(rmse_holt/n_wind)
rmse_hw = np.sqrt(rmse_hw/n_wind)

#Initialise Results data frame
res = pd.DataFrame()
res['h'] = pd.Series(range(1,hmax+1))
res['SES'] = pd.Series(rmse_ses)
res['Holt'] = pd.Series(rmse_holt)
res['HW'] = pd.Series(rmse_hw)

print(res)
```

- The best method one-step ahead is the Holt Winters Method.
- The best method twelve-steps ahead is Simple Exponential Smoothing.
- Holt performs reasonably well at short horizons but very poorly at medium to long horizons.
- Another thing to note is that we are not using all of the data. If we use all available data then we will have some windows towards the end for which longer-horizon forecasts can not be evaluated. This is not a major problem, but note that the denominator in MSE will be different for different horizons (unlike here where we could divide by 192 at all horizons to compute RMSE).

7. What is a major shortcoming of this evaluation? Hint: What happens on April 25th in Australia.

April 25th is Anzac Day a major public holiday in Australia. In 2021 it was on a Sunday meaning the holiday was moved to the 26th. Therefore the evaluation period includes a public holiday and these days are typically idiosyncratic.

1.2 Additional Exercises (For those who finish quickly or as subsequent homework)

Modify the code above:

1. To use a rolling rather than expanding window.
2. To roll the window forward by 4 hours rather than half an hour.
3. To use the mean absolute error $MAE = \frac{1}{192}\sum|y+t+h-\hat{y}_{t+h}|$ as an evaluation criterion.