HW3

Charles Dotson

September 21, 2022

Contents

```
[314]: '''Importing Packages'''
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
from IPython.display import Markdown as md
import statsmodels.tsa.stattools as ts
import statsmodels.api as sm
import datetime
from loess import loess_1d
from statsmodels.graphics.tsaplots import plot_acf
from openpyxl import Workbook, load_workbook
from sklearn import linear_model
from statsmodels.tsa.ar_model import AutoReg, ar_select_order
from scipy.linalg import toeplitz
import math
import scipy.stats as stats
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
%matplotlib inline
```

1 Exercise 3.7 in the text: The quality of Pinot Noir wine is thought to be related to the properties of clarity, aroma, body, flavor, and oakiness. Data for 38 wines are given in Table E3.4.

```
[315]: '''Importing data'''
Pinot_data = pd.read_excel('PinotNoir.xlsx')
```

1.1 Fit a multiple linear regression model relating wine quality to these predictors. Do not include the "Region" variable in the model.

```
[316]: Pinot_n_region = Pinot_data.drop(['Region'], axis=1)

X = Pinot_n_region[['x1','x2','x3','x4','x5']]
y = Pinot_n_region['y']

X = sm.add_constant(X)
est = sm.OLS(y, X).fit()
summary = est.summary(title='Regression Results Pinot (Table 1)')
```

1.2 Test for significance of regression. What conclusions can you draw?

```
[318]: md(summary.as_latex())
```

[318]:

Dep. Variable:	y	**R-squared:**	0.721
Model:	OLS	**Adj. R-squared:**	0.677
Method:	Least Squares	**F-statistic:**	16.51
Date:	Tue, 20 Sep 2022	**Prob (F-statistic):**	4.70e-08
Time:	23:54:28	**Log-Likelihood:**	-56.378
No. Observations:	38	**AIC:**	124.8
Df Residuals:	32	**BIC:**	134.6
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	3.9969	2.232	1.791	0.083	-0.549	8.543
x1	2.3395	1.735	1.349	0.187	-1.194	5.873
x2	0.4826	0.272	1.771	0.086	-0.072	1.038
x3	0.2732	0.333	0.821	0.418	-0.404	0.951
x4	1.1683	0.304	3.837	0.001	0.548	1.789
x5	-0.6840	0.271	-2.522	0.017	-1.236	-0.132

Omnibus:	1.181	**Durbin-Watson:**	0.837
Prob(Omnibus):	0.554	**Jarque-Bera (JB):**	1.020
Skew:	-0.384	**Prob(JB):**	0.601
Kurtosis:	2.770	**Cond. No.**	134.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

From Table 1 above we can see that the F-Stat probability shows signifigance but the R^2 does not show the greatest fit. We can see that DW test is closer to 0 and thus shows that there is a posibility of auto correlation of the residuals. We can also see that the JB test is showing normality.

1.3 Use t-tests to assess the contribution of each predictor to the model. Discuss your findings.

[319]:
```
t_stats = []
p_values = []
for i in Pinot_n_region.columns.tolist()[:-1]:
    x_ = Pinot_n_region[i]
    ttest = stats.ttest_1samp(x_, popmean=0)
    t_stats.append(ttest.statistic)
    p_values.append(ttest.pvalue)

beta_t_test_df = pd.DataFrame(index =
    [r'$\beta_1$',r'$\beta_2$',r'$\beta_3$',r'$\beta_4$', r'$\beta_5$'],
    columns= ['stastistic', 'p-values']
    )
beta_t_test_df['stastistic'] = t_stats
beta_t_test_df['p-values'] = p_values
md('''
{}
```

```
{}

'''.format(r'$$\text{Table 2}$$',beta_t_test_df.to_markdown()))
```

[319]:

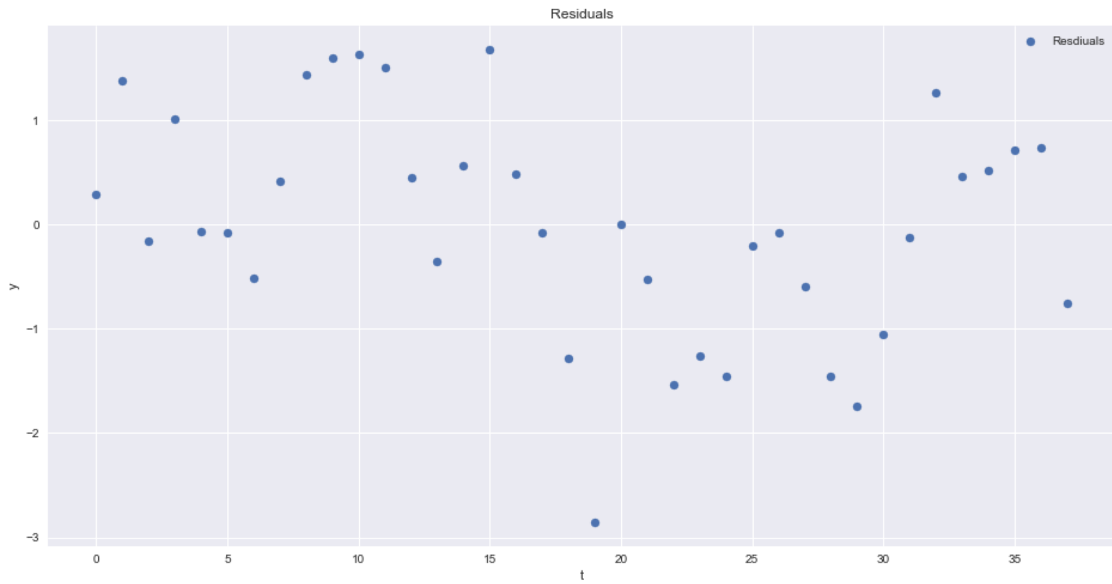
$$\text{Table 2}$$

	stastistic	p-values
β_1	45.9283	3.11551e-34
β_2	27.6045	2.749e-26
β_3	35.0371	5.57985e-30
β_4	28.5935	7.91129e-27
β_5	35.5418	3.33646e-30

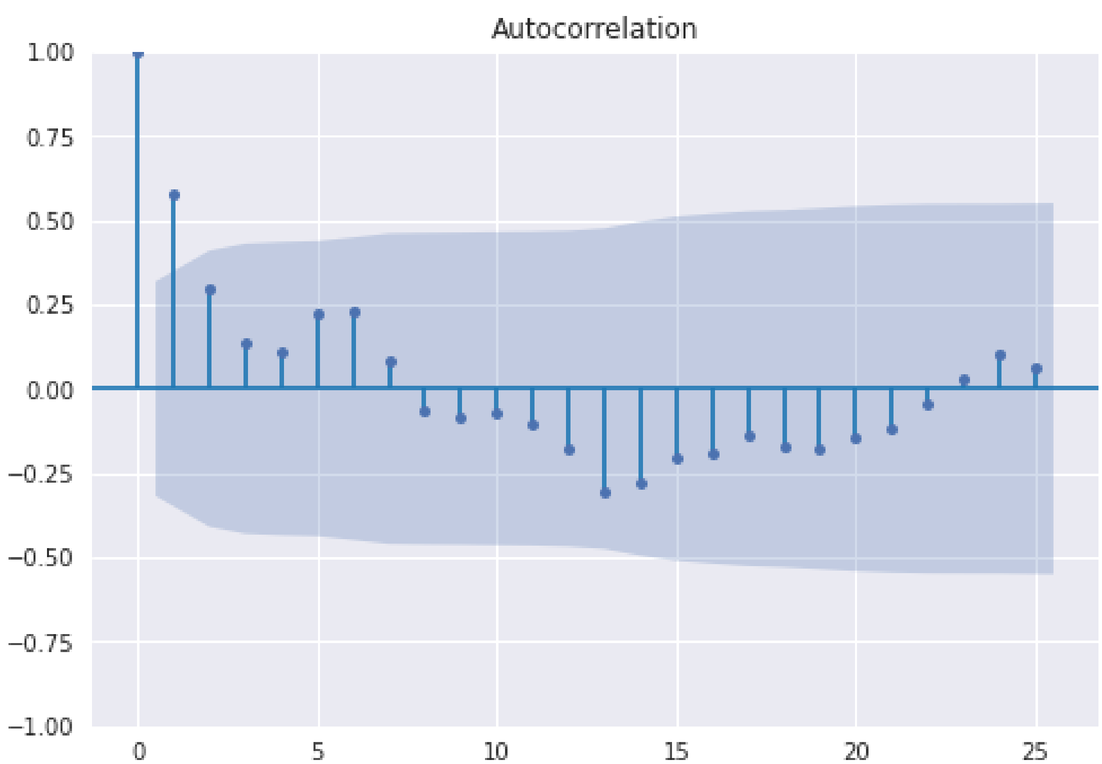
Here are saying that $H_0 : \beta_i = 0$ and $H_1 : \beta_i \neq 0$. From the table above we can see all coefficients and thus variables have a significant effect on the model.

1.4 Analyze the residuals from this model. Is the model adequate?

[320]:
```
with plt.style.context('seaborn'):
    fig = plt.figure(figsize=(16,8))
    ax = plt.axes()
    plt.scatter([i for i in range(len(Pinot_data))],est.resid, label =␣
 ↪'Resdiuals')
    ax.set_xlabel('t')
    ax.set_ylabel('y')
    plt.title('Residuals')
    plt.legend()
    plt.show()
```



[321]:
```
with plt.style.context('seaborn'):
    fig = plot_acf(est.resid, lags=25)
```



From Table 1 we can see that that JB test shows possible normality but there seems to be auto-correlation in the DW test yet in the ACF generated above it would shows decay which would lead us to beleive that there is non.

1.5 Calculate R^2 and the adjusted R^2 for this model. Compare these values to the R^2 and adjusted R^2 for the linear regression model relating wine quality to only the predictors "Aroma" and "Flavor." Discuss your results.

[322]:
```
'''Fitting the secondary model proposed'''
X = Pinot_n_region[['x2','x4']]
y = Pinot_n_region['y']

X = sm.add_constant(X)
est_2 = sm.OLS(y, X).fit()
summary = est_2.summary(title='Regression Results Pinot Aroma and Flavor (Table␣
↪3)')
```

[323]:
```
md(summary.as_latex())
```

[323]:

Dep. Variable:	y	**R-squared:**	0.659
Model:	OLS	**Adj. R-squared:**	0.639
Method:	Least Squares	**F-statistic:**	33.75
Date:	Tue, 20 Sep 2022	**Prob (F-statistic):**	6.81e-09
Time:	23:54:29	**Log-Likelihood:**	-60.188
No. Observations:	38	**AIC:**	126.4
Df Residuals:	35	**BIC:**	131.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P> |t|**	**[0.025**	**0.975]**
const	4.3462	1.009	4.307	0.000	2.298	6.395
x2	0.5180	0.276	1.877	0.069	-0.042	1.078
x4	1.1702	0.291	4.027	0.000	0.580	1.760

Omnibus:	0.321	**Durbin-Watson:**	0.869
Prob(Omnibus):	0.852	**Jarque-Bera (JB):**	0.499
Skew:	0.076	**Prob(JB):**	0.779
Kurtosis:	2.460	**Cond. No.**	35.8

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Refer to Table 1 and Table 2.

We can see from both tables the model using more predictors has a higher R^2 and adjusted R^2 than that of the one with only two predictor variables. This is most likely due to the fact that the Oakiness variables which was left out had a better p-value than Aroma. Thus we are removing more insignificant predictor variables but also taking a significant predictor with it. Also leaving a non-significant predictor varibale in the training data.

1.6 Find a 95% CI for the regression coefficient for "Flavor" for both models in part e. Discuss any differences.

We can see from table 1 and 2 that the 95% CI for Flavor is $(0.548, 1.789)$ and $(0.580, 1.760)$ respectively.

2 Given a process $y_T = \beta_0 + \beta_1 t + \epsilon_t$, $\epsilon_t \overset{uncorr.}{\sim} (0, \sigma^2)$. Show that the second-order exponentially smoothed estimate, $\hat{y}_T = 2\tilde{y}_T^{(1)} - \tilde{y}_T^{(2)}$ (as in the equation (4.23) of the text), is an unbiased estimate of $\mathbb{E}(y_T)$.

We are asked to show $\hat{y}_T$ is unbiased estimate of $\mathbb{E}[y_T]$. To do this we will show $\mathbb{E}[\hat{y}_T] = \mathbb{E}[y_T]$.

$$
\begin{aligned}
\mathbb{E}[\hat{y}_T] &= 2\mathbb{E}[\tilde{y}_T^{(1)}] - \mathbb{E}[\tilde{y}_T^{(2)}] \\
&= 2\mathbb{E}[\tilde{y}_T^{(1)}] - \mathbb{E}[\tilde{y}_T^{(1)}] + \frac{1-\lambda}{\lambda}\beta_1 \\
&= \mathbb{E}[\tilde{y}_T^{(1)}] + \frac{1-\lambda}{\lambda}\beta_1 \\
&= \mathbb{E}[y_T] - \frac{1-\lambda}{\lambda}\beta_1 + \frac{1-\lambda}{\lambda}\beta_1 \\
\mathbb{E}[\hat{y}_T] &= \mathbb{E}[y_T]
\end{aligned}
$$

This shows that the expectation of the exponentially smoothed process is an unbiaed estimate of the given process.

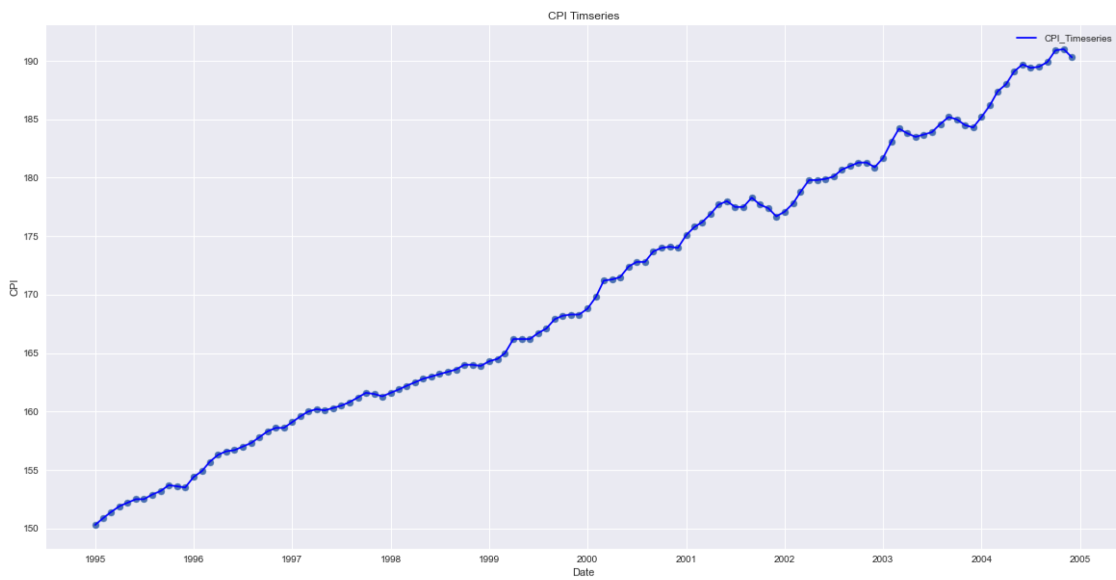
3 (Example 4.2) Referring to the dataset, CPI.xlsx, posted in HW03 link on coursesite,

[324]:
```python
'''importing data'''

CPI_data = pd.read_excel('CPI.xlsx')

'''Fixing Data'''
df_0 = CPI_data[['Month', 'CPI']]
df_0 = df_0.set_index('Month')
df_1 = CPI_data[['Month.1', 'CPI.1']]
df_1.columns = ['Month', 'CPI']
df_1 = df_1.set_index('Month')
```

```
df_2 = CPI_data[['Month.2', 'CPI.2']]
df_2.columns = ['Month', 'CPI']
df_2 = df_2.set_index('Month')
df_3 = CPI_data[['Month.3', 'CPI.3']]
df_3.columns = ['Month', 'CPI']
df_3 = df_3.set_index('Month')
df_4 = CPI_data[['Month.4', 'CPI.4']]
df_4.columns = ['Month', 'CPI']
df_4 = df_4.set_index('Month')

CPI_data_cleaned = pd.concat([df_0,df_1,df_2,df_3,df_4])
```

3.1 Visualize the series by generating the time-series plot for "CPI" series that is to be indexed by "Month"series. Comment on whether there exists a linear trend.

[326]:
```
with plt.style.context('seaborn'):
    fig = plt.figure(figsize=(20,10))
    ax = plt.axes()
    plt.plot(CPI_data_cleaned, c = 'Blue', label = 'CPI_Timeseries')
    plt.scatter(CPI_data_cleaned.index, CPI_data_cleaned['CPI'])
    ax.set_xlabel('Date')
    ax.set_ylabel('CPI')
    plt.title('CPI Timseries')
    plt.legend()
    plt.show()
```

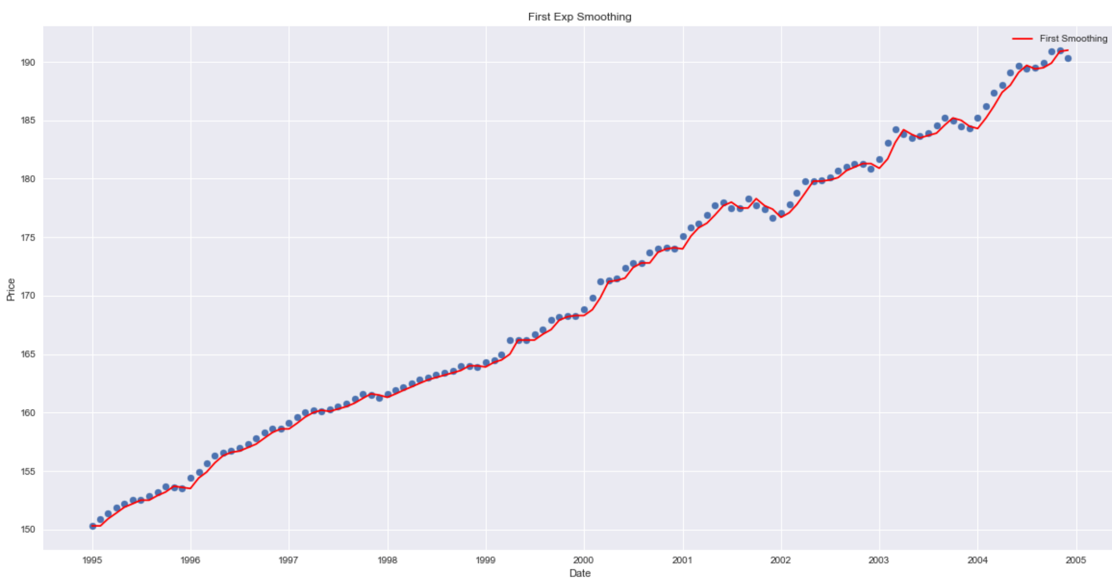


3.1.1 Comments:

There seems to be a positive linear trend.

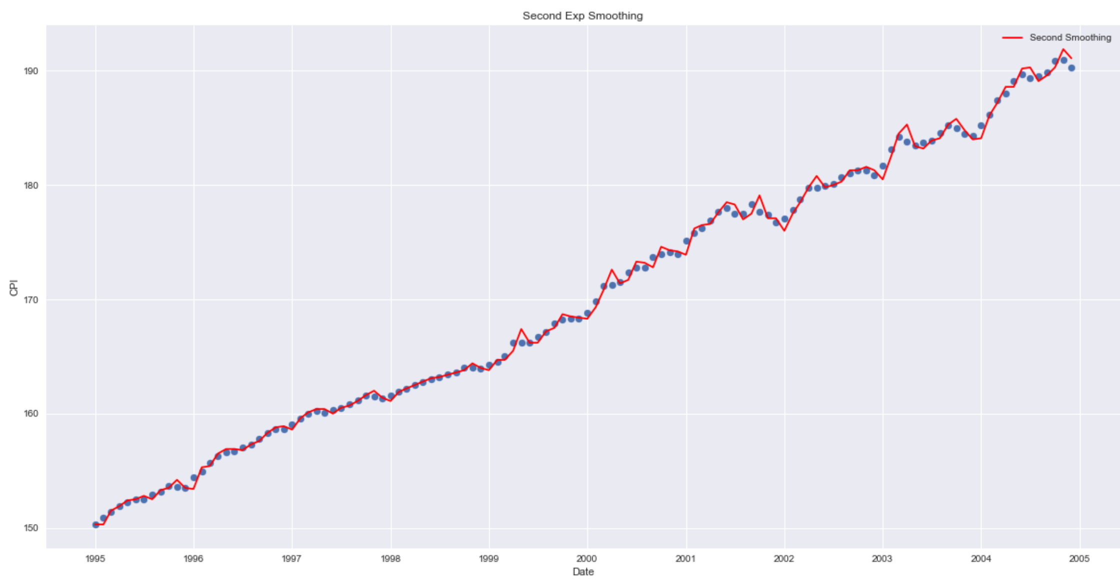
3.2 Find and plot the first-order exponentially smoothed estimates (using $\tilde{y}_0^{(1)} = y_1, \lambda = 0.3$) overlapped with the original series. Comment on the fitness of the first-order exponential smoother on the CPI series.

[327]:
```
fit1 = ExponentialSmoothing(CPI_data_cleaned.values,␣
↪initialization_method="heuristic", use_boxcox = 0.3).fit()
CPI_data_cleaned_fitted = CPI_data_cleaned
CPI_data_cleaned_fitted['fitted'] = fit1.fittedvalues
with plt.style.context('seaborn'):
    fig = plt.figure(figsize=(20,10))
    ax = plt.axes()
    plt.plot(CPI_data_cleaned_fitted['fitted'], c = 'Red', label = 'First␣
↪Smoothing')
    plt.scatter(CPI_data_cleaned.index, CPI_data_cleaned['CPI'])
    ax.set_xlabel('Date')
    ax.set_ylabel('Price')
    plt.legend()
    plt.title('First Exp Smoothing')
    plt.show()
```



3.3 Find and plot the second-order exponentially smoothed estimates (using $\tilde{y}_0^{(1)} = y_1, \tilde{y}_0^{(2)} = \tilde{y}_0^{(1)}, \lambda = 0.3$) overlapped with the original series. Comment on the fitness of the first-order exponential smoother on the CPI series.

[328]:
```
fit2 = ExponentialSmoothing(fit1.fittedvalues,␣
↪initialization_method="heuristic", use_boxcox = 0.3).fit()

CPI_data_cleaned_fitted['fitted2'] = 2*fit1.fittedvalues - fit2.fittedvalues
with plt.style.context('seaborn'):
    fig = plt.figure(figsize=(20,10))
    ax = plt.axes()
    plt.plot(CPI_data_cleaned_fitted['fitted2'], c = 'Red', label = 'Second␣
↪Smoothing')
    plt.scatter(CPI_data_cleaned.index, CPI_data_cleaned['CPI'])
    ax.set_xlabel('Date')
    ax.set_ylabel('CPI')
    plt.legend()
    plt.title('Second Exp Smoothing')
    plt.show()
```



3.3.1 Comments

It seems to capture the process better yet over estimates the positive variances when varainces increases in the process.

3.4 Implement the second-order exponential smoother to incorporate (4.24) as a new function. Fit the function to the data with keeping $\lambda = 0.3$ and plot the new fit with original data. Comment on the comparison between this revised fit to the fit in (c).