Advanced Research Methods – Assignment 1

Time series modelling & forecast case study

In this assignment, we will conduct a case study: modelling some real time series data, with a variety of models, then make a forecast into the future.

The modelling is split into 2 parts;

1. ARMA model and forecast
2. Neural network-based model and forecast

The submissions for this assignment are the code used to produce the analysis, models and forecast (including figures), and a report describing what you have done, why you have taken the approach you have and what you have interpreted.

Learning outcomes:

- Be able to take existing methods for modelling and analysing data
- Suggesting improvements to the modelling to develop the understanding and compare with traditional methods
- Write a report about your findings

We will be modelling two types of data from two companies.

1. Sales data from a company called Johnson & Johnson. They want you to model the sales data and make a forecast for the next 24 months.

   We will us the company sales data in the file jj.csv

2. The online retail company Amazon's closing share price data.

   This can be found in AMZN.csv.

The supplementary information below includes an outline of the approach to take when performing this case study. It builds on work done in the tutorials.

Supplementary information:

Part 1: ARMA model and forecast

Let us start by setting up the notebook:

```
from statsmodels.graphics.tsaplots import plot_pacf
```

```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
from sklearn.metrics import mean_squared_error
from tqdm import tqdm_notebook
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from itertools import product
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
plt.rcParams['figure.figsize'] = [10, 7.5]

# read in data file
data = pd.read_csv('jj.csv')
data.head()
```

Remember, if you are given a new function to use, it is a good idea to look at the function docs, e.g:

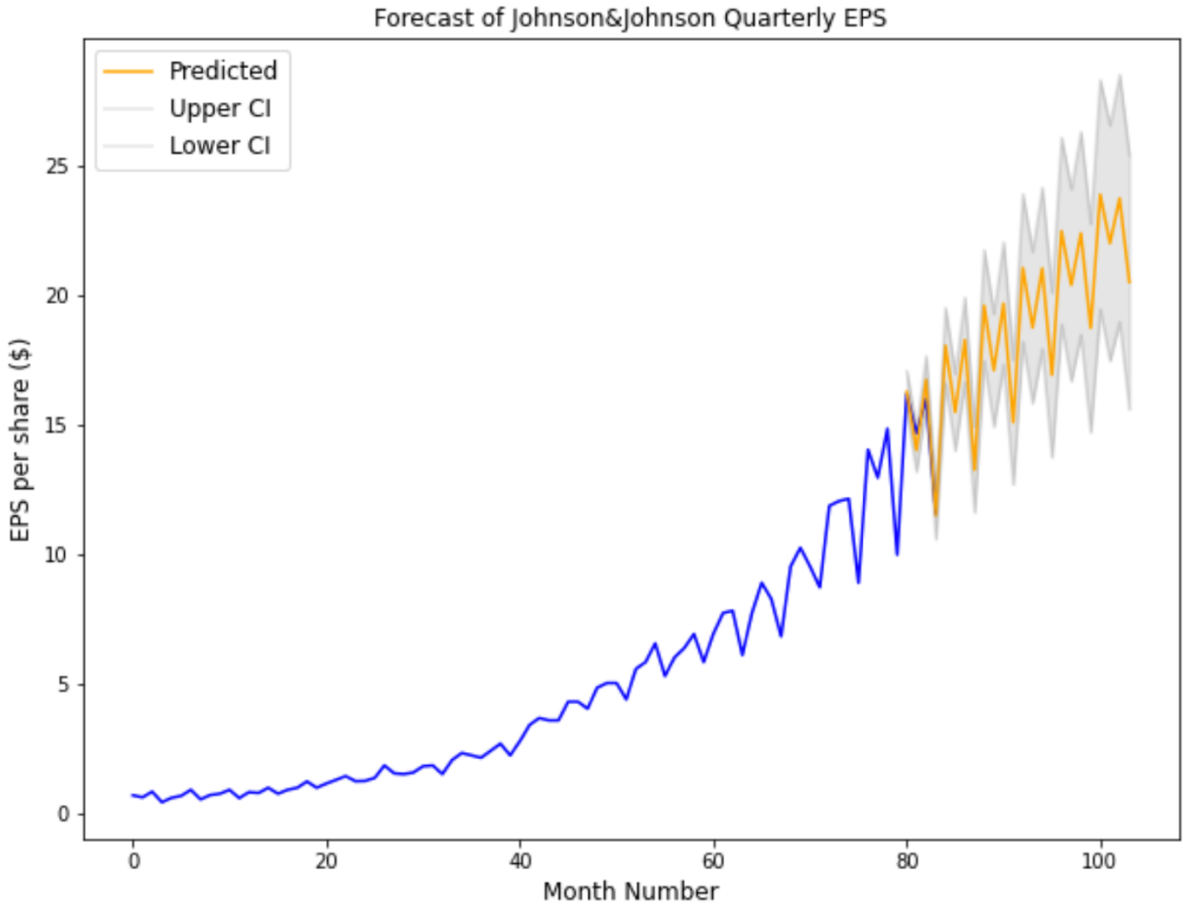
https://www.statsmodels.org/dev/generated/statsmodels.tsa.arima_process.ArmaProcess.html

We may also want to us `pmdarima`. If it is not already installed, you can do e.g:

```
pip install pmdarima
import pmdarima as pm
```

The docs here: https://pypi.org/project/pmdarima/

At the end of the exercise, we are aiming for a plot which looks similar to this:



Here is a recommended approach to completing the task:

1. Read in the data and make a plot

2. Investigate the properties of the data

3. What does the ACF and PACF look like?

4. Perform an ADF test for stationarity. Is it stationary?

5. If it's not stationary, take the log and use the differencing operator to make it stationary.
   a. Try using the Box-Cox transformation (unit 1) to find the optimal transformation

6. Find the best ARMA(p,q) model fit

I would like you to demonstrate 2 approaches for doing this:

a. Write a function to loop over every p,d,q from zero up to e.g. p=8,q=8, d=1, find the best model fit then record the AIC value for each model variant. Then find the model variant with the lowest AIC.

```
from statsmodels.tsa.arima.model import ARIMA
```

Then loop over all p,d,q combinations…

loop….

```
model = ARIMA(train, order=(p,d,q))
```

record AIC
….end loop

b. Use an automated best model fit finder, such as pmdarima:

```
from statsmodels.tsa.arima_model import ARIMA
import pmdarima as pm
```

Do you get the same answer as before?

7. Once you have your best model orders, define the new model and fit to the data and inspect the summary information.

```
model_fit = model.fit()
print(model_fit.summary())
model_fit.plot_diagnostics()
```

8. Produce a forecast for the next 24 months. We can use `get_prediction()` or `get_forecast()`

e.g:

```
n_forecast = 24
predict = model_fit.get_prediction(end=model.nobs +
n_forecast)
idx = np.arange(len(predict.predicted_mean))
fig, ax = plt.subplots()
ax.plot(data['data'], 'blue')
ax.plot(idx[-n_forecast:], predict.predicted_mean[-
n_forecast:], 'k--')
ax.set(title='Forecast of Johnson&Johnson Sales)
plt.show()
```

9. Add confidence intervals to the forecast using get_forecast()

```
predictions_int = model_fit.get_forecast(steps=24)
predictions_int.predicted_mean
```

---

10. Investigate the components of the data using Fourier transforms (e.g. the DFT or FFT).

Using the code discussed in Unit 4, make the Power Spectral Density (periodogram / spectrogram) of the supplied time series.

Does this indicate any patterns in the data?