What is Auto Regression?

Within Time Series Analysis there are several algorithms that computers can use to predict future values. One of these algorithms is the auto regression algorithm. The auto regression model is characterized by using previous values to make predictions of future values. For example in the stock market auto regression is used to predict the rise in stocks based on the stocks historic values.

Stock Example

To explain further we could consider a stock with the value of $5 dollars per share. Using the auto regression model the values $1, 3, and 4$ are put into the model along with the current value of $5. For this example 1$ is three months ago, $3 is two months ago, and 4$ is one month ago with $5 being the present price. In order to calculate this the values would be placed in the algorithm below.

$$m_t = \beta_0 + \beta_1 m_{t-1} + \beta_2 m_{t-2} + \beta_3 m_{t-3} + \epsilon_t$$

The result of this algorithm would be the predicted value of the stock.

Auto Regression in Raven

For the purposes of Raven, auto regression works similarly to wavelet, and fourier transformation. The auto regression algorithm takes signals from the reactor as input. Then, based on this input, the algorithm characterizes the input and makes a prediction of expected signals in the future. This adds a furhter layer to protection to the reactor and further applicability to Raven.

Code

Below is code related to the auto regression algorithm and a demonstration of its functionality. For demonstration purposes statsmodel is being used to predict the price of gamestop.

In [18]:
```
!pip install statsmodels
!pip install matplotlib

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.tsa.stattools as sm
import statsmodels.graphics.tsaplots as sp
import warnings
warnings.filterwarnings('ignore')
```

```
Requirement already satisfied: statsmodels in c:\users\admin\miniconda3\lib\site-packages (0.12.2)
Requirement already satisfied: scipy>=1.1 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.6.2)
Requirement already satisfied: patsy>=0.5 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (0.5.1)
Requirement already satisfied: pandas>=0.21 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.2.4)
Requirement already satisfied: numpy>=1.15 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.20.2)
Requirement already satisfied: numpy>=1.15 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.20.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\admin\miniconda3\lib\site-packages (from pandas>=0.21->statsmodels) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\admin\miniconda3\lib\site-packages (from pandas>=0.21->statsmodels) (2021.1)
Requirement already satisfied: six in c:\users\admin\miniconda3\lib\site-packages (from patsy>=0.5->statsmodels) (1.15.0)
Requirement already satisfied: numpy>=1.15 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.20.2)
Requirement already satisfied: six in c:\users\admin\miniconda3\lib\site-packages (from patsy>=0.5->statsmodels) (1.15.0)
Requirement already satisfied: numpy>=1.15 in c:\users\admin\miniconda3\lib\site-packages (from statsmodels) (1.20.2)
Requirement already satisfied: matplotlib in c:\users\admin\miniconda3\lib\site-packages (3.4.1)
Requirement already satisfied: numpy>=1.16 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (1.20.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (8.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (2.8.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\admin\miniconda3\lib\site-packages (from matplotlib) (2.4.7)
Requirement already satisfied: six in c:\users\admin\miniconda3\lib\site-packages (from cycler>=0.10->matplotlib) (1.15.0)
Requirement already satisfied: six in c:\users\admin\miniconda3\lib\site-packages (from cycler>=0.10->matplotlib) (1.15.0)
```

In [19]:
```
df = pd.read_csv("GME.csv")
# df
```

In [20]:
```
df = pd.read_csv("gameStopInput.csv")
# df
```

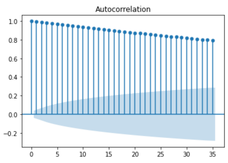
In [21]:
```
array = sm.acf(df.y[1:2600], nlags=26000)
# array
```

In [22]:
```
xAfter=[]
yAfter=[]
for i in range(len(array)):
    xAfter.append(i)
    yAfter.append(array[i])
```

In [23]:
```
x = np.array(xAfter)
y = np.array(yAfter)
dataset = pd.DataFrame({'x': x, 'y': y}, columns=['x', 'y'])
# dataset
```

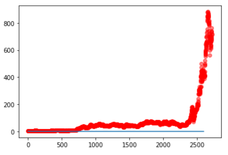
The graph below shows the autocorrelation factor that is produced from the gamestop data.

In [24]:
```
sp.plot_acf(array)
plt.show()
```



The chart below shows the gamestop stock data next to the autocorrelation results. Gamestop data is shown in red and autocorrelation results are shown in blue.

In [25]:
```
x = df.x
y = df.y

dat = pd.DataFrame({'x': x, 'y': y})
dat.to_csv('gameStop.csv', index=False)


plt.plot(x, y, 'o', alpha=0.5, color='red')
plt.plot(dataset.x, dataset.y)
plt.show()
```



Below is a second example of Auto Regression. This time predictions for the Tesla stock price is being used.

Auto Regression in Raven

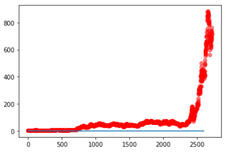
This is a visual representation of what Raven is doing with the same dataset, Gamestop.

In [26]:
```
df = pd.read_csv("gameStopInput.csv")
```

In [27]:
```
dataset = pd.read_csv("GameStopResults.csv")
# dataset
```

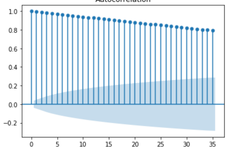
As shown before here is the gamestop stock data next to the produced autocorrelation values.

In [28]:
```
x = df.x
y = df.y


dat = pd.DataFrame({'x': x, 'y': y})
dat.to_csv('gameStop.csv', index=False)


plt.plot(x, y, 'o', alpha=0.5, color='red')
plt.plot(dataset.x, dataset.y)
plt.show()
```



The above graph shows the autocorrelation compared to the price of gamestop. This is not a very good representation of autocorrelation however, a better description can be found in the graph below.

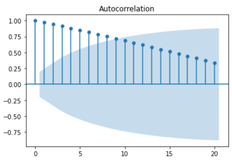
In [29]:
```
sp.plot_acf(dataset.y)
plt.show()
```



The graph above depicts autocorrelation in a more understandable way. If we adjust the parameters to more specific instances such as from 2500 to 2600 the autocorrelation values will also change for example the slope may be steeper or more gradual depending on the prices within the 2500 to 2600 range.

In [30]:
```
array = sm.acf(dataset.y[1:100], nlags=100)
# array
```

In [32]:
```
sp.plot_acf(array)
plt.show()
```



As we can see with a y range from 1 to 100 the slope of autocorrelation becomes much steeper.