ARIMA

```
# fit an ARIMA model and plot residual errors
import pandas as pd
from pandas import datetime
from pandas import read_csv
from pandas import DataFrame
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt


inputfile = "../Datasets/TECO2.2000.2021.csv"


ts = pd.read_csv(inputfile, header=0, index_col=0, squeeze=True)
ts.fechaHora = pd.to_datetime(ts.fechaHora)
ts.fechaHora=pd.to_datetime(ts.fechaHora).dt.date
ts.fechaHora=pd.DatetimeIndex(ts.fechaHora)
ts=ts.sort_index(ascending=False)

# fit model
model = ARIMA(ts.ultimoPrecio.values, order=(5,1,0))
model_fit = model.fit()
# summary of fit model
print(model_fit.summary())
# line plot of residuals
residuals = DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
# density plot of residuals
residuals.plot(kind='kde')
pyplot.show()
# summary stats of residuals
print(residuals.describe())


# evaluate an ARIMA model using a walk-forward validation
X = ts.ultimoPrecio.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
# walk-forward validation
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit()
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
```

```python
    obs = test[t]
    history.append(obs)
    #print('predicted=%f, expected=%f' % (yhat, obs))

# evaluate forecasts
rmse = sqrt(mean_squared_error(test, predictions))
print('Test RMSE: %.3f' % rmse)
# plot forecasts against actual outcomes
pyplot.plot(test)
pyplot.plot(predictions, color='red')
pyplot.show()
```