Abhishek Patwardhan

D17A - 57

ADS Experiment 7

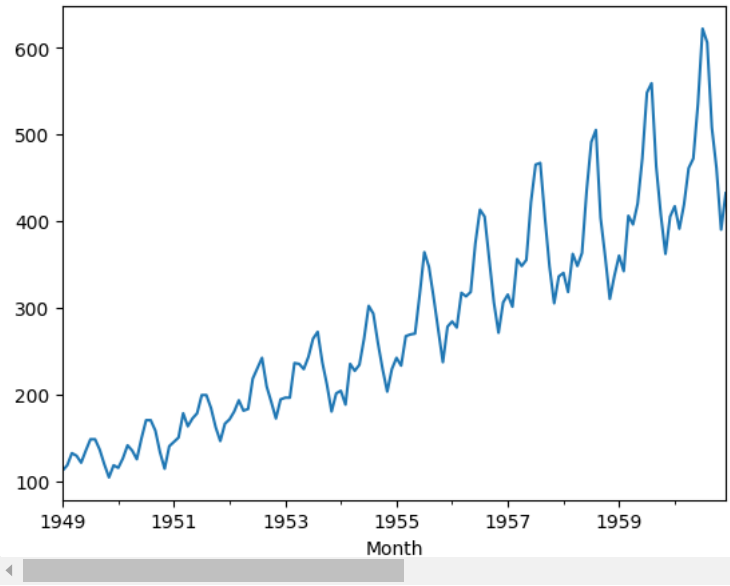
==> Loading the basic libraries

```
from pandas import read_csv
from pandas import datetime
from pandas import DataFrame
from matplotlib import pyplot

series = read_csv('AirPassengers.csv', header=0, parse_dates=[0], index_col=0, squeeze=True)
print(series.head())
series.plot()
pyplot.show()
```

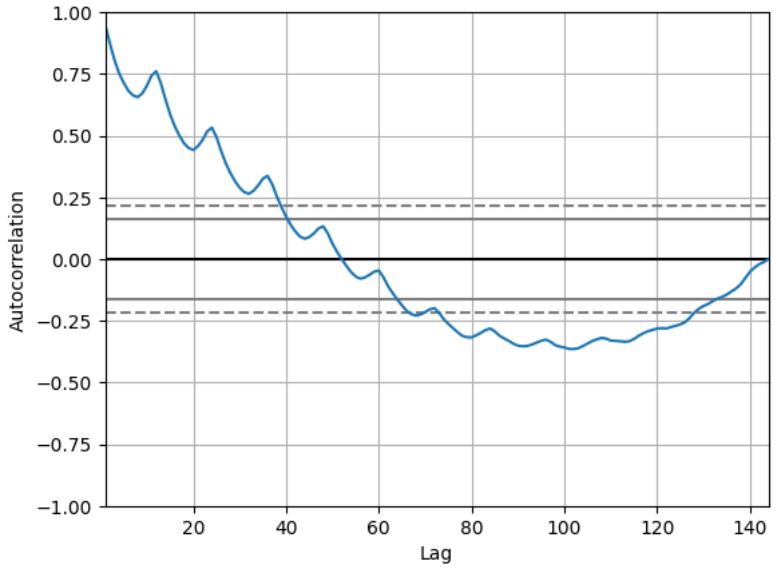
```
<ipython-input-11-4716754fad24>:2: FutureWarning: The pandas.datetime class is deprec
  from pandas import datetime
<ipython-input-11-4716754fad24>:6: FutureWarning: The squeeze argument has been depre


  series = read_csv('AirPassengers.csv', header=0, parse_dates=[0], index_col=0, sque
Month
1949-01-01    112
1949-02-01    118
1949-03-01    132
1949-04-01    129
1949-05-01    121
Name: Passengers, dtype: int64
```



==> Autocorrelation plot of the time series

```
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(series)
pyplot.show()
```




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


# Fit Model
model = ARIMA(series, order=(5,1,0))
model_fit = model.fit()
```

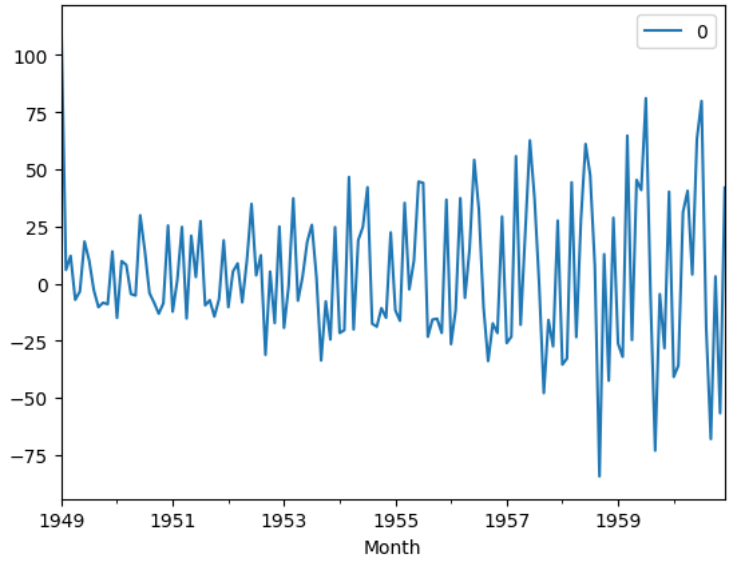
```
/usr/local/lib/python3.9/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided,
  self._init_dates(dates, freq)
/usr/local/lib/python3.9/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided,
  self._init_dates(dates, freq)
/usr/local/lib/python3.9/dist-packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency information was provided,
  self._init_dates(dates, freq)
```

```
# summary of fit model
print(model_fit.summary())
```

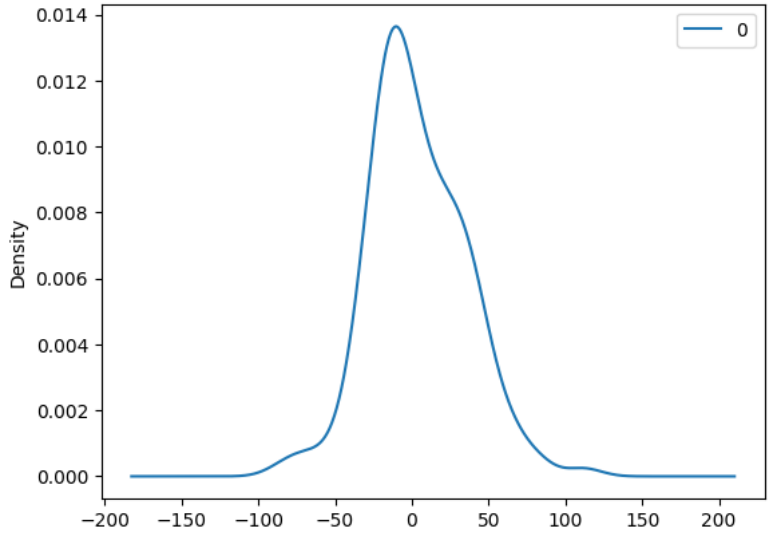
```
                               SARIMAX Results
==============================================================================
Dep. Variable:             Passengers   No. Observations:                  144
Model:                 ARIMA(5, 1, 0)   Log Likelihood                -689.067
Date:                Sun, 09 Apr 2023   AIC                           1390.135
Time:                        15:26:48   BIC                           1407.912
Sample:                    01-01-1949   HQIC                          1397.358
                         - 12-01-1960
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3223      0.097      3.334      0.001       0.133       0.512
ar.L2         -0.2170      0.078     -2.776      0.006      -0.370      -0.064
ar.L3         -0.0646      0.071     -0.915      0.360      -0.203       0.074
ar.L4         -0.2641      0.075     -3.519      0.000      -0.411      -0.117
ar.L5          0.0250      0.094      0.267      0.790      -0.159       0.209
sigma2       893.7229    113.383      7.882      0.000     671.497    1115.949
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 0.45
Prob(Q):                              0.97   Prob(JB):                         0.80
Heteroskedasticity (H):               8.15   Skew:                             0.12
Prob(H) (two-sided):                  0.00   Kurtosis:                         3.14
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

```
# 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())
```

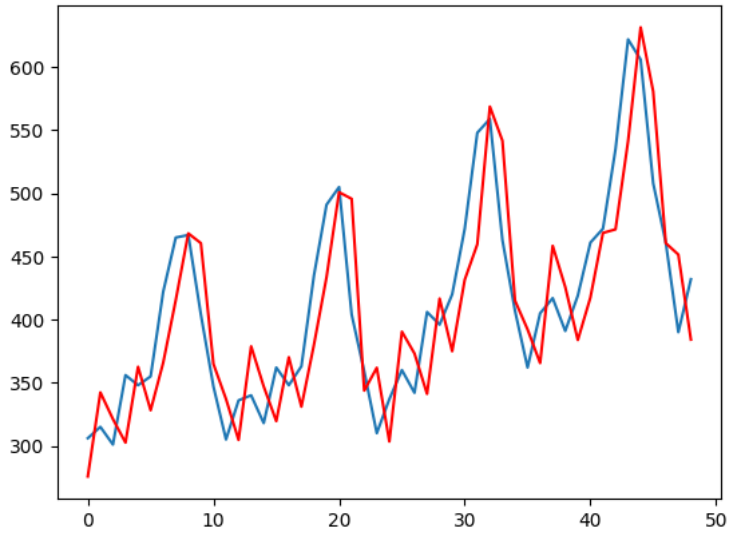
```
                0
count  144.000000
mean     3.893151
std     31.087159
min    -84.397612
25%    -16.548937
50%     -1.792613
75%     25.066151
max    112.000000
```

```
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt

# split into train and test sets
X = series.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)
 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()
```

```
Test RMSE: 44.465
```



Colab paid products - Cancel contracts here

✓ 8s completed at 10:47 PM