README.md

Arima

Autoregressive integrated moving model is the generalized moving average model for time series predictions. A non season Arima has three components p, d, q.

- p - Specifies the order of time lag.
- d - Specifies the degree of differencing
- q - Specifies order of moving average.

ARIMA is implemented python stats library which will be used for training and predictions. This project uses a non seasonal variant of ARIMA.

Data set

Non seasonal ARIMA has been verified against two data sets. The first one includes temperature data and second one includes passenger data. Both are available online.
Kaggle Passenger Data
Temperature in Melbourne

The task is to predict the future time series value for both the data sets.

Utility methods

```
def isSeriesStationary(series):
    pValue = adfuller(series)[1]
    if pValue > 0.05:
        return False
    else:
        return True

def isSeriesStationaryAvg(series, delta = 2):
    split = int(len(series)/2)
    split1, split2 = series[:split], series[split:]
    avg1, avg2 = split1.mean(), split2.mean()
    var1, var2 = split1.var(), split2.var()
    if abs(avg1 - avg2) > delta or abs(var1 - var2) > delta**2:
        return False
    else:
        return True
```

For non seasonal Arima to work it is necessary that time series is stationary. This means that the time series is devoid of trend and seasonality and its average and standard deviation remains constant throughout time. The above two methods aids in calculating the same. The first method performs adfuller test and checks for P value. This method is very reliable in checking the stationarity of time series. Second method is taking a look at average. This may not be very reliable. It is advised to perform adfuller test for testing stationarity.

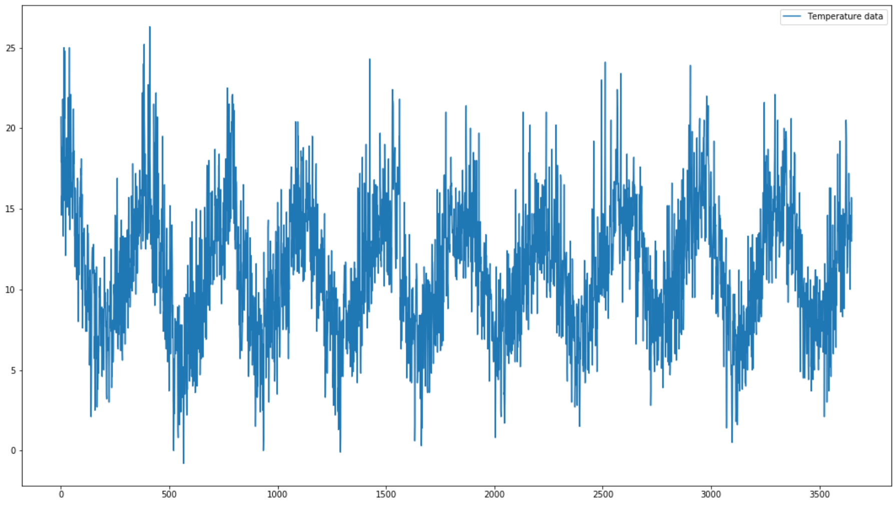
```
def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return np.array(diff)

def inverse_difference(history, yhat, interval=1):
    return yhat + history[-interval]

def describeSeries(data, label):
    fig = matplotlib.pyplot.gcf()
    fig.set_size_inches(18.5, 10.5)
    plt.plot(data, label = "Series")
    plt.plot(pd.rolling_mean(data, window = 2), '--', label = "Rolling mean")
    plt.plot(pd.rolling_std(data, 2), ":", label = "Rolling Std")
    plt.legend()
    plt.savefig(label)
    plt.clf()
```

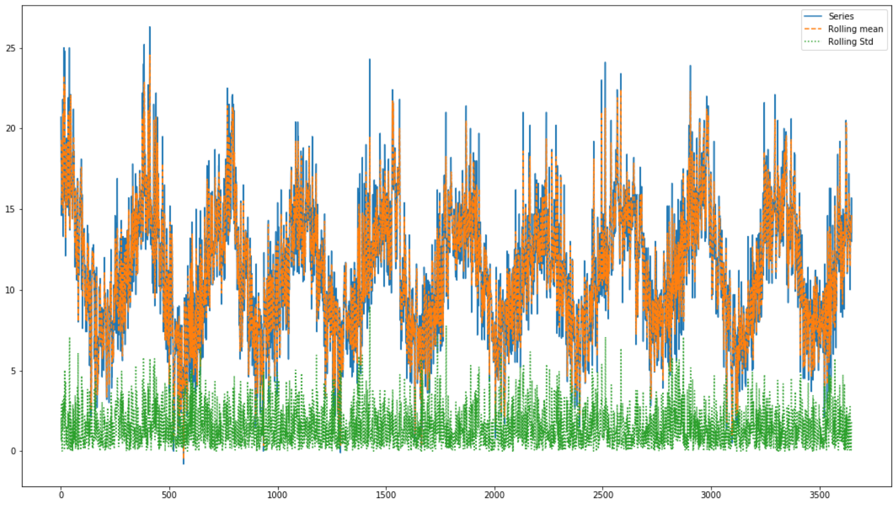
The method describeSeries creates a plot of time series with rolling mean and rolling standard deviation. Other two methods are used for differencing and performing inverse differentiation of the time series. These methods are used to make time series stationary.

```
def splitTrainTest(series, testSplit):
    totalData = len(series)
    trainSplit = int(totalData * (1 - testSplit))
    trainSet = series[:trainSplit]
    testSet = series[trainSplit:]
    return trainSet, testSet
```

The above method is used to split time series in train and test data. Note the data in time series is not shuffled.

ARIMA for temperature dataset



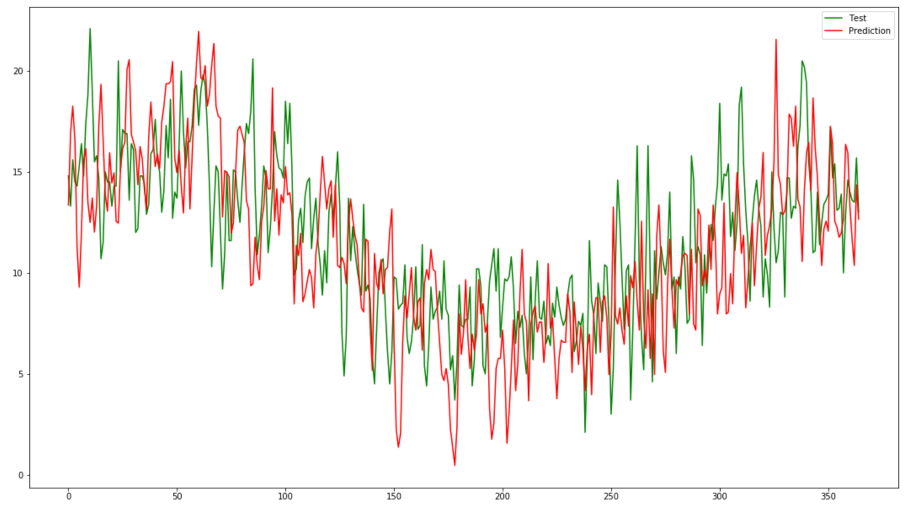


After performing the stationarity test both the methods return that the time series is stationary. So no preprocessing is required, however still time series differentiation has been performed. The lag value is 365. The reason for this is that the series is daily temperature data (365 days).

```
trainSet, testSet = splitTrainTest(temperatureData["Temperature"].values, 0.1)

differencedTrainSet = difference(trainSet, 365)
model = ARIMA(differencedTrainSet, order=(7,0,1))
"""Fit model with non constant trend and no displacement"""
model_fit = model.fit(disp = 0)
forecast = model_fit.predict(len(differencedTrainSet),
          len(differencedTrainSet) + len(testSet))

yPrediction = []
history = list(trainSet)
for f in forecast:
    yPredict = inverse_difference(history, f, 365)
    yPrediction.append(yPredict)
    history.append(yPredict)
```

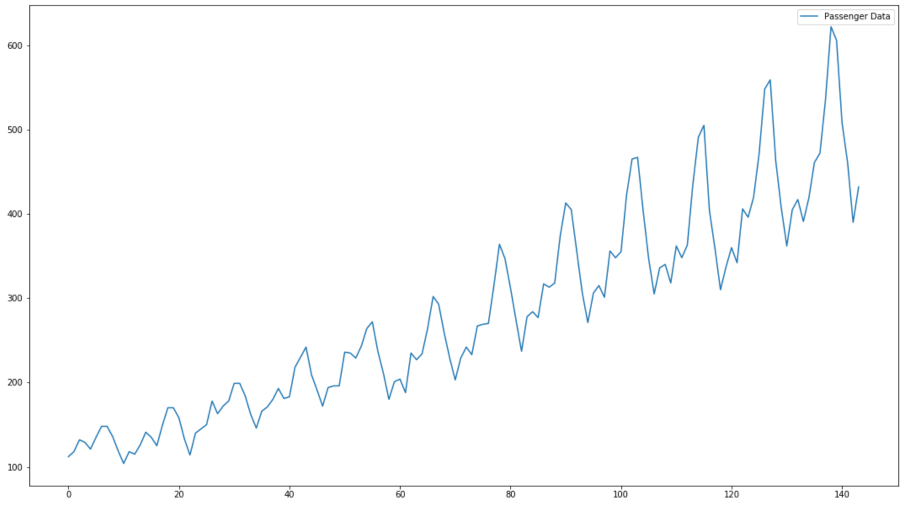
ARMA model is trained with time lag of 7 as no of days in week. No degree of differencing required as the series is already differenced. Moving average is 1. No intuition for this setting.

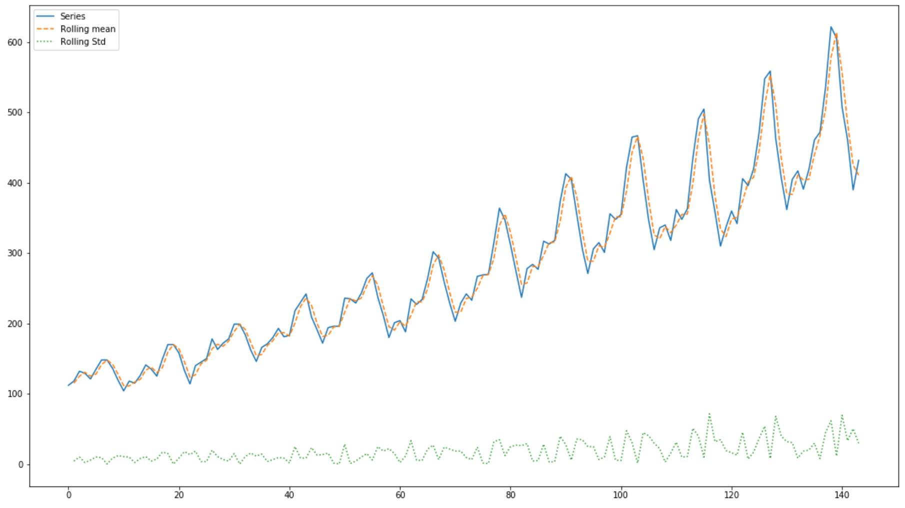
Prediction plot



The RMSE for this data set is 3.65

Arima for passenger Dataset



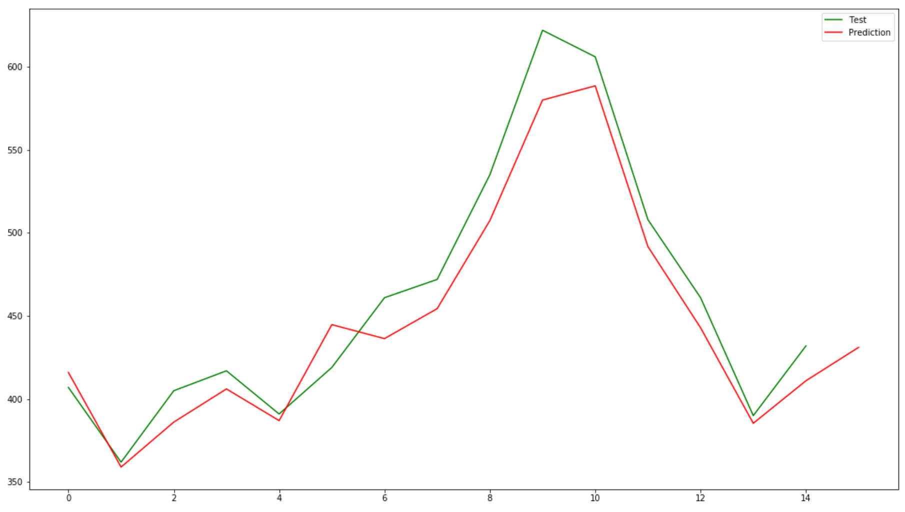


It is evident that the series is not stationary. Both the test indicate that the time series is not stationary. After differencing with lag of 12 the time series becomes stationary. The data includes monthly passenger data for airlines. Hence the lag of 12 was taken.

```
trainSet, testSet = splitTrainTest(passengersData['#Passengers'], 0.1)

differencedTrainSet = difference(trainSet, 12) * 1.0

model = ARIMA(differencedTrainSet, order=(1,0,1))
"""Fit model with non constant trend and no displacement"""
model_fit = model.fit(disp = 0)
forecast = model_fit.predict(len(differencedTrainSet),
len(differencedTrainSet) + len(testSet))


yPrediction = []
history = list(trainSet)
for f in forecast:
    yPredict = inverse_difference(history, f, 12)
    yPrediction.append(yPredict)
    history.append(yPredict)
```

The model is trained with additional lag of 1 and moving average of 1. There is no intuition for this setting. Different configurations were tried and this one yields best result.

Prediction plot



The RMSE for this dataset is 18.92