Time Series Analysis

Project III

By: Rohith Lingala,

Harshith Kasthuri,

Charan Reddy Mannuru,

James Notoma,

Josh Kolesar

Guided by-

Dr. John Miller
CSCI 6360- Data Science II

Nov 16th, 2024

Introduction:

In this report we will be focusing on a few Time series algorithms and discussing their performance on well known dataset Covid_19 by forecasting their results on response variables i.e. new_deaths.

TimeSeries.scala program forecasts COVID-19 data using three different time-series models:

1. **Random Walk Model (RW)**
2. **Auto-Regressive Model of order 1 (AR1)**
3. **Auto-Regressive Model of order 2 (AR2)**

It involves loading the data, defining response variables, creating models, training them, testing them, and performing validations.

Loading Data :

1. **Source File:**
    - The data is read from `covid_19_weekly.csv`.
2. **Headers:**
    - The column names in the dataset (e.g., `new_cases`, `new_deaths`) are listed in the `header` array.
3. **Response Variable:**
    - The variable to be predicted, e.g., `new_deaths`, is defined as `response`.
4. **Methods:**
    - `loadData`:
        - Loads selected columns (`x_strs`) as exogenous variables and one column (`y_str`) as the response variable.
        - Returns a `MatrixD` for predictors and a `VectorD` for the response.
        - Useful for building models with independent variables.
    - `loadData_y`:
        - Loads only the response variable (`new_deaths`) as a `VectorD`.
        - Useful for univariate forecasting.

- `loadData_yy`:
  - Loads multiple columns as a `MatrixD`.
  - Useful for multivariate models like VAR.

5. **Data Trimming:**
   - The `trim` parameter allows skipping initial rows (e.g., where values are all zeros) to clean the data.

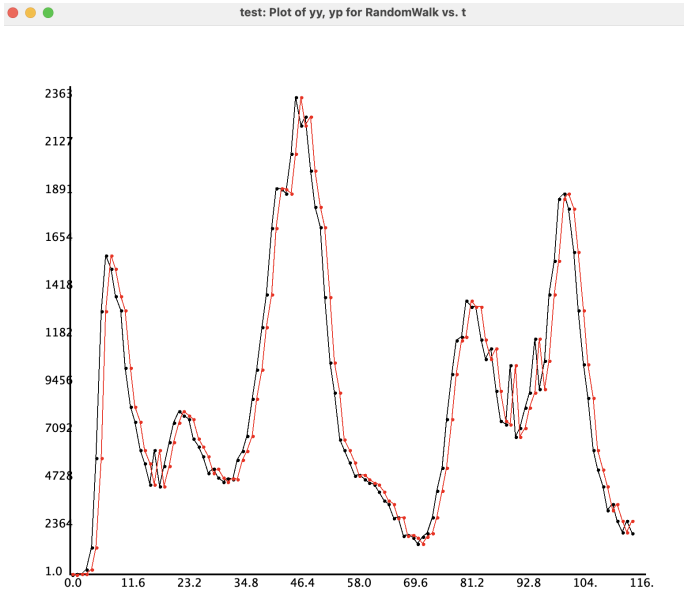
Random Walk:

In Random walk, R2 value in In-sample is more than the train and test R2 whose values are 0.906 and 0.888 respectively and below images shows the prediction for hyperparmeter 1.

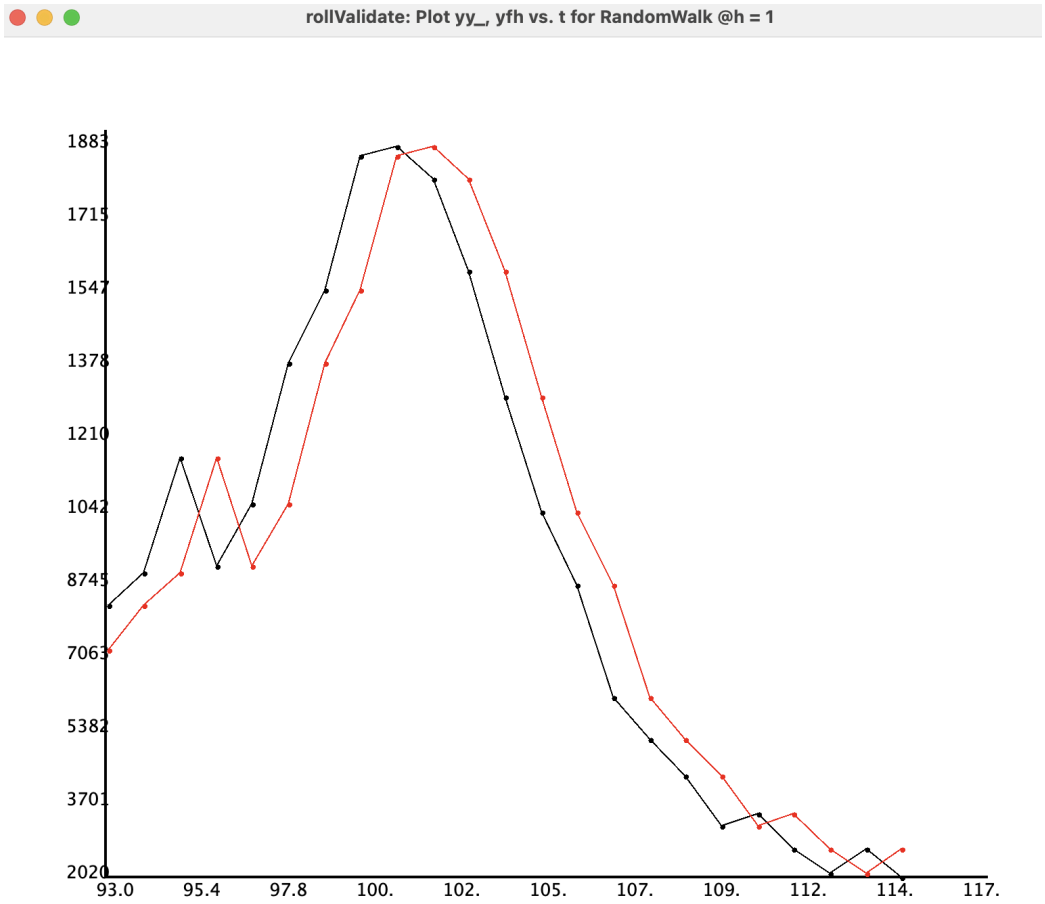
Insample:

rSq -> VectorD(0.906288, 0.713537, 0.441865, 0.114432, -0.212585, -0.512304)

rSqBar -> VectorD(0.905466, 0.711001, 0.436882, 0.106454, -0.223609, -0.526179)



Train & Test:



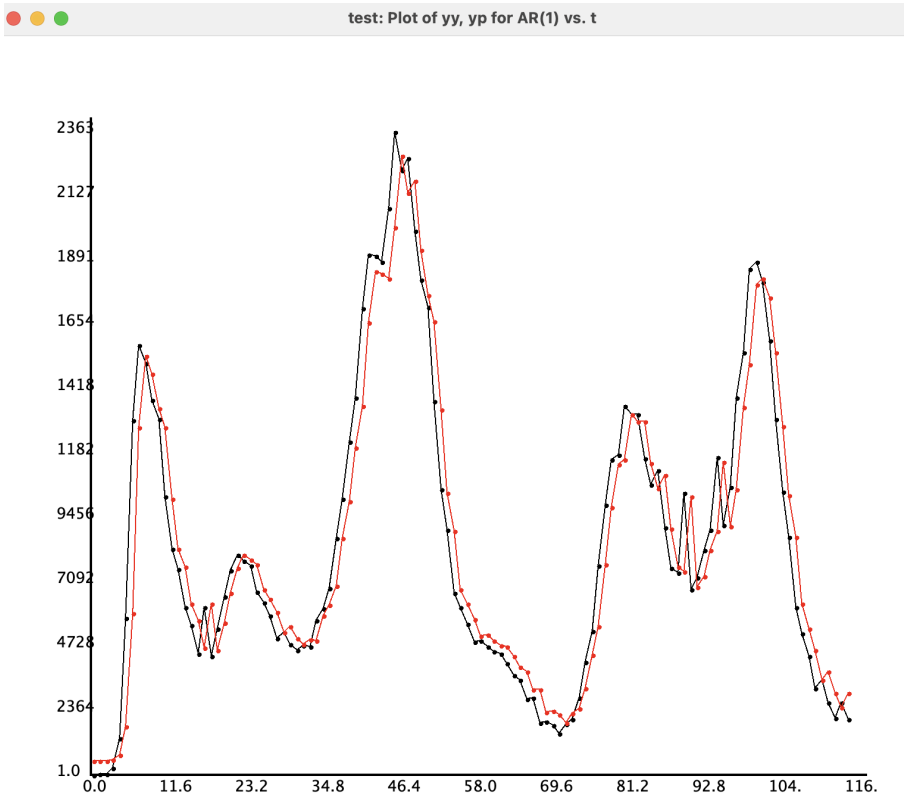
rSq -> VectorD(0.893076, 0.684745, 0.388791, 0.0480943, -0.304726, -0.660500)

rSqBar -> VectorD(0.888216, 0.669733, 0.358230, -0.00200605, -0.377211, -0.758177)

AR1 :

Same as Randomwalk Insample performed better that in Train and test and the R2 values were improved compared to Randomwalk.

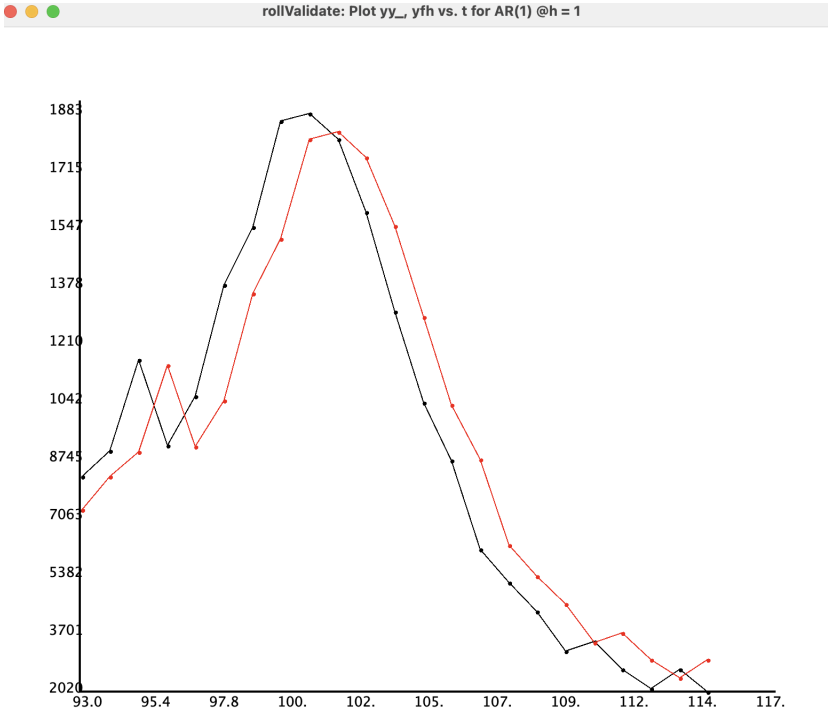
InSample:



rSq -> VectorD(0.910664, 0.740794, 0.521500, 0.277842, 0.0422287, -0.163902)

rSqBar -> VectorD(0.909887, 0.738520, 0.517265, 0.271394, 0.0336002, -0.174483)

Train and test



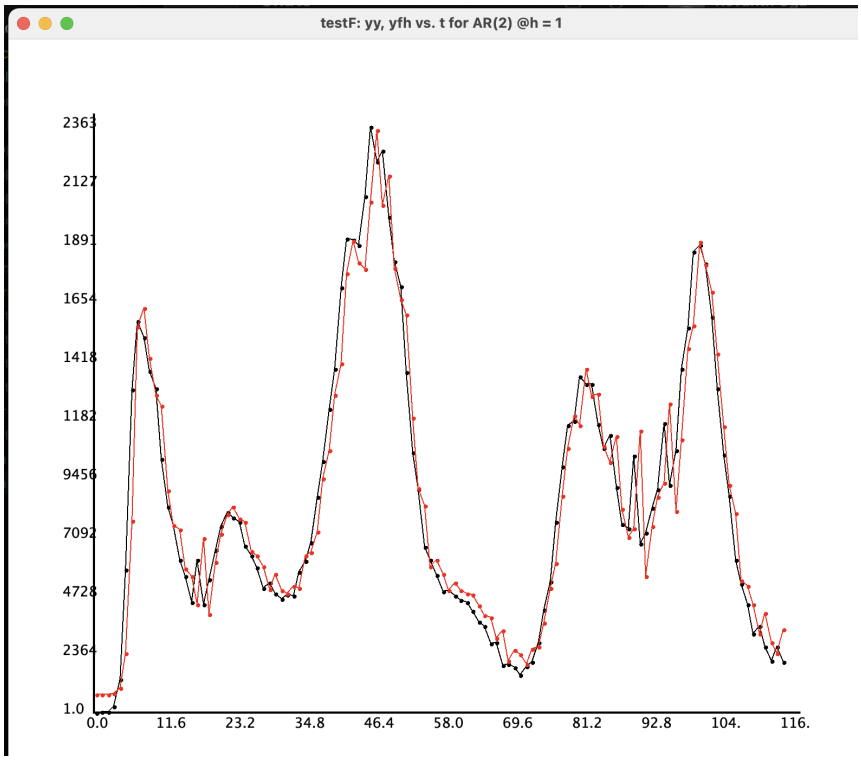
rSq -> VectorD(0.892962, 0.696079, 0.433779, 0.157531, -0.106624, -0.350869)

rSqBar -> VectorD(0.888097, 0.681607, 0.405468, 0.113190, -0.168104, -0.430332)

AR2:

AR2 performed way better than AR1 and Randomwalk whose R2 values for Insample and train and test are 0.932 and 0.914 respectively.

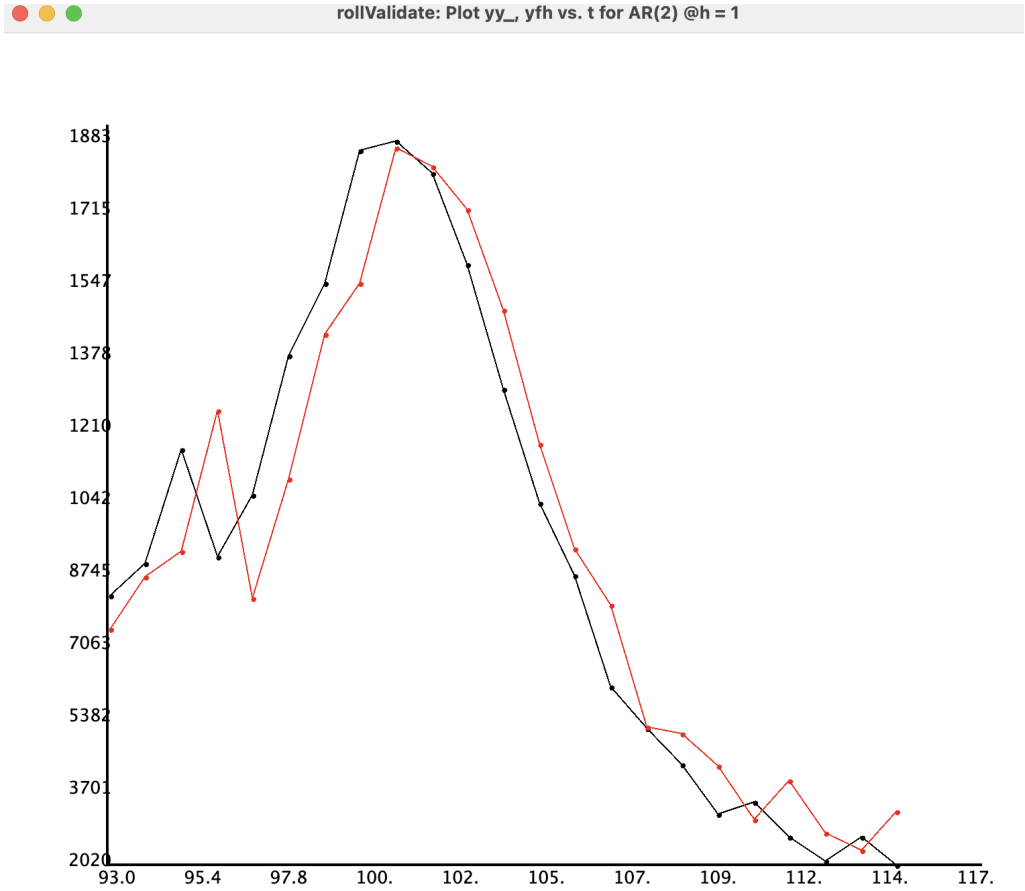
Insample :



rSq -> VectorD(0.932725, 0.803068, 0.625478, 0.424202, 0.254051, 0.114079)

rSqBar -> VectorD(0.931534, 0.799552, 0.618730, 0.413733, 0.240364, 0.0976726)

Train and test :



rSq -> VectorD(0.914804, 0.760847, 0.543322, 0.285391, 0.0635117, -0.146414)

rSqBar -> VectorD(0.906691, 0.736932, 0.495250, 0.205990, -0.0466634, -0.289715)

Comparison Table:

Test method Model	In-Sample	Train and Test
Random Walk	0.906288	0.893076
AR1	0.910664	0.892962
AR2	0.932725	0.914804

Clearly AR2 dominated both Randomwalk and AR1 in both In-sample and Train and test.