Introduction

Time series analysis is the process of using statistical techniques to model and explain a time-dependent series of data points.

Time series forecasting is the process of using a model to generate predictions (forecasts) for future events based on known past events.

Time series data has a natural temporal ordering this differs from typical data mining/machine learning applications where each data point is an independent example of the concept to be learned, and the ordering of data points within a data set does not matter.

Examples of time series applications include:

- capacity planning.
- sales forecasting and future staffing levels.

Requirements

The Weka time series modeling environment requires Weka >= 3.7.3 and is provided as a package that can be installed via the package manager.

Any of Weka's regression algorithms can be applied to learn a model.

Once installed via the package manager.

- The time series modeling environment can be found in a new tab in Weka's Explorer GUI
- Data is brought into the environment in the normal manner by loading from a file
- The environment has both basic and advanced configuration options

Basic Configuration

Example Dataset: airline.arff

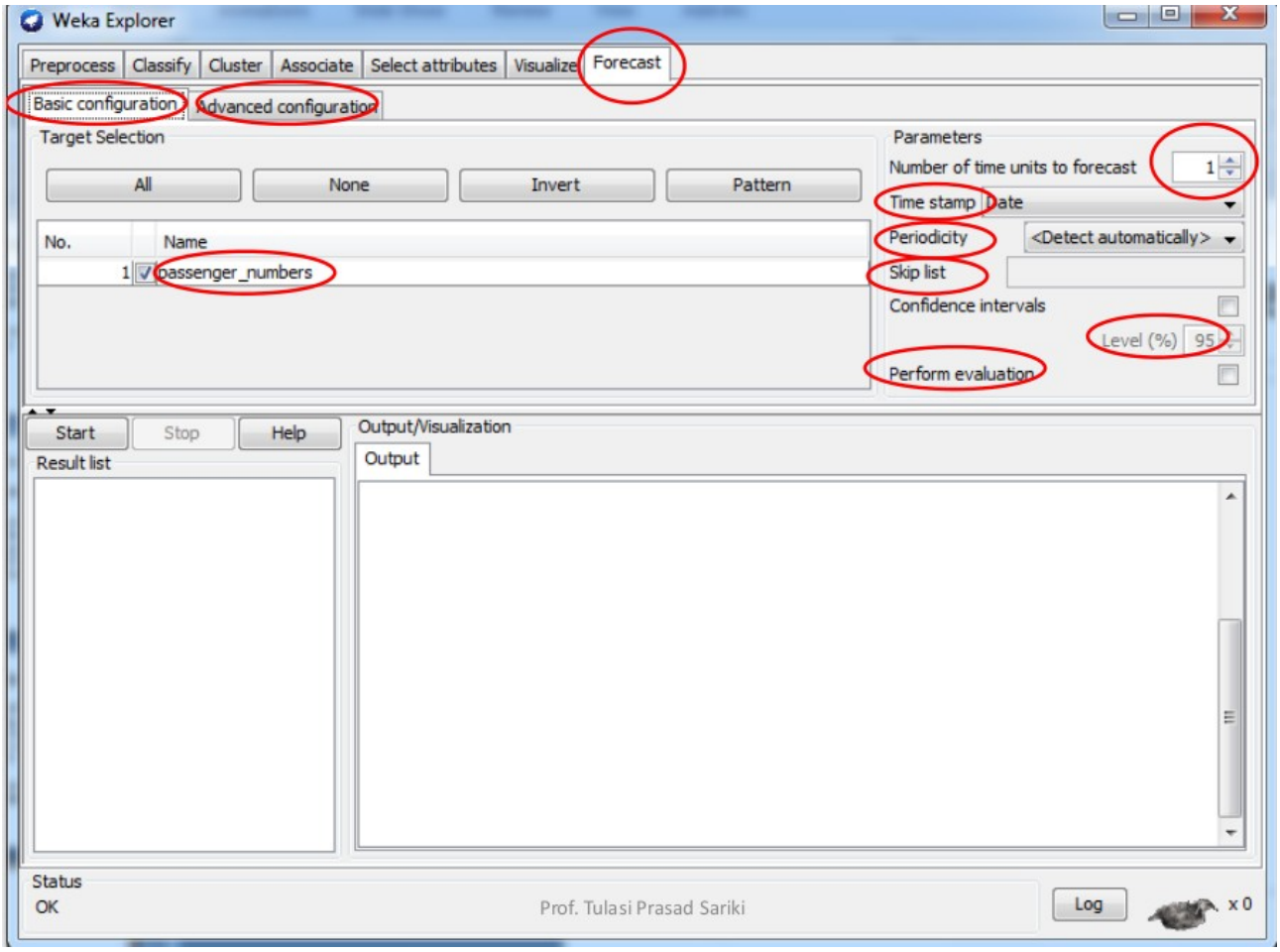
- That has one series of data: monthly passenger numbers for an airline for the years 1949-60. Aside from the passenger numbers, the data also includes a date time stamp.
- The basic configuration panel automatically selects the single target series and the "Date" time stamp field.
- Different Parameters in the Parameters section(top right-hand side)
    - user can enter the number of time units to forecast beyond the end of the supplied data.
    - there is a drop-down box for specifying the periodicity of the data (Date is Default).

- Skip List allows the user to specify time periods that should not count as a time stamp increment with respect to the modeling.
  - "weekend", "sat", "tuesday", "mar" & "october"
  - specific dates (with optional formatting string) such as "2011-07-04@yyyy-MM-dd",

Different Parameters

Confidence intervals

- Default is 95%, a 95% confidence level means that 95% of the true target values fell within the interval.
- Perform Evaluation
  - The Advanced Configuration panel allows the user to fine tune configuration by selecting which metrics to compute and whether to hold-out some data from the end of the training data as a separate test set.



Advanced Configuration

• Base learner

- provides control over which Weka learning algorithm is used to model the time series.
- It also allows the user to configure parameters specific to the learning algorithm selected.

• Lag creation

- allows the user to control and manipulate how lagged variables are created
- Ex: if you had monthly sales data then including lags up to 12 time steps into the past would make sense; for hourly data, you might want lags up to 24 time steps or perhaps 12.

• Periodic attributes

- allows the user to customize which date-derived periodic attributes are created.

• Overlay data

- allows the user to specify fields (if any) that should be considered as "overlay" data.
- That is, data that is not to be forecasted. The default is not to use overlay data.

• Evaluation

- Allows the user to select which evaluation metrics they wish to see, and configure whether to evaluate using the training data and/or a set of data held out from the end of the training data.

• Output provides options that control what textual and graphical output are produced by the system

Available Metrics

• Mean absolute error (MAE): sum(abs(predicted - actual)) / N

• Mean squared error (MSE): sum((predicted - actual)^2) / N

• Root mean squared error (RMSE): sqrt(sum((predicted - actual)^2) / N)

• Mean absolute percentage error (MAPE): sum(abs((predicted - actual) / actual)) / N

• Direction accuracy (DAC): count(sign(actual_current - actual_previous)

• Relative absolute error (RAE): sum(abs(predicted - actual)) / sum(abs(previous_target - actual))

• Root relative squared error (RRSE):

sqrt(sum((predicted - actual)^2) / N) / sqrt(sum(previous_target - actual)^2) / N)