Reserve Bank of New Zealand
Analytical Notes

Using Job Transitions Data As A Labour Market Indicator

AN2020/02

Christopher Ball, Adam Richardson and Thomas van Florenstein Mulder

January 2020

Reserve Bank of New Zealand Analytical Note Series
ISSN 2230-5505

Reserve Bank of New Zealand
PO Box 2498
Wellington
NEW ZEALAND

www.rbnz.govt.nz

RESERVE BANK
OF NEW ZEALAND
TE PŪTEA MATUA

Technical Appendix

A. Models

A.1. Autoregressive model (AR)

As a benchmark model we use a univariate AR model of order 1 for the quarterly unemployment rate, u_t:

$$u_t = a_0 + a_1 u_{t-1} + e_t$$

Where a_0 and a_1 are the estimated parameters and e_t is the residual term.

A.2. Autoregressive integrated moving average model (ARIMA)

As a second benchmark we use an ARIMA(p, d, q) model to forecast unemployment rates, where p is the order of autoregression, d is the order of integration and q is the moving average order. To determine d we use an augmented Dickey-Fuller test, for p and q we use an Akaike information criterion. The ARIMA therefore takes the following form:

$$u_t = a_0 + a_1 u_{t-1} + \cdots + a_p u_{t-p} + \theta_1 e_t + \cdots + \theta_q e_{t-q}$$

Where $u_t = \Delta U_t$ if the order of integration is one.

A.3. Random Walk

The third benchmark model is a random walk with no drift, this takes the following form,

$$u_t = u_{t-1} + e_t$$

Simply put the nowcast for the unemployment rate is equal to the unemployment rate of the quarter before.

A.3. Boosted Trees

Boosted trees is an ensemble method used to build a high quality predictor out of a number of lower quality regression trees. This algorithm starts by fitting an initial regression tree to the target variable guided by the minimisation of the least squares. We then take the residuals of the initial model and fit another regression tree to the residuals, these two trees are combined to create a new prediction of the target variable. We then calculate the residuals of the new prediction and continue the process as outlined until the number of regression trees is equal to the pre-specified number of learning cycles. We are essentially iterating through multiple steps of the following:

$$F_m(x) = F_{m-1}(x) + v\Delta_m(x)$$

Where $F_m(x)$ is the new model mapping x to the target variable, $F_{m-1}(x)$ is the previous model, $\Delta_m(x)$ is the new learner created from the residuals of the $F_{m-1}(x)$, and Δ is a shrinkage parameter (learning rate). The learning rate is used as a form of regularisation to minimise the amount the model is overfitting in-sample.

We use the $LS_{Tree-Boost}$ algorithm described in Friedman (2001). In this setup we have four hyper parameters. These include the learning rate, the number of trees to include in the model, the characteristics of the individual trees (we focus on the number of terminal nodes, in line with Friedman (2001)) and lastly the number of features to include. We largely follow the simulations provided by Friedman and set the learning rate to 0.1, the number of trees to 1000, the number of terminal nodes to 11 and we use all the features of our data set.

A.4. Lasso, Ridge and Elastic Net (ENET)

Ridge regression is a linear methodology used to create a parsimonious model when the number of features exceeds the number of observations. Ridge regression shrinks the coefficients by imposing a penalty on their size. This is known as L2 regularisation and helps to deal with multicollinearity and decreasing model complexity. However in this form of regression none of the coefficients are removed completely therefore, this model is not used to create sparsity. The ridge regression takes the following form:

$$\beta = argmin[\sum_{i=1}^{l}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2]$$

Where λ is the hyperparameter that determines the severity of shrinkage of the parameters. We set $\lambda = 0.8$ to ensure the models ability to generalise to unseen data.

Lasso regression uses L1 regularisation to create sparse models. Lasso creates a penalty equal to the magnitude of the coefficients. Given a sufficiently large λ, coefficients can become zero and the lasso acts as a feature selector. The lasso regression takes the following form:

$$\beta = argmin[\sum_{i=1}^{l}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|]$$

Where $\lambda = 0.4$ so as to minimise the chance of overfitting.

Elastic net regression is a combination of the lasso and the ridge regression and therefore the elastic net can shrink coefficients or remove them completely. The elastic net penalty is therefore a convex sum of the lasso and ridge penalty terms and has the following form.

$$\beta = argmin[\sum_{i=1}^{l}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p}(\alpha|\beta_j| + (1-\alpha)\beta_j^2)]$$

We set α to 0.8 to reflect the better performance of the lasso regression and we set λ to 0.4 as in the lasso case.

A.5. Support Vector Regression (SVR)

Support vector regression is a non-parametric approach which, under epsilon-insensitivity, aims to find a function, $f(x)$, that has at maximum ε deviation from the observed target variables, y_i of the training set. The literature notes that a function which is both smooth and has at maximum ε errors may not exist, therefore we introduce slack variables, ξ_i, that measure and assign cost to errors larger than ε. We set any errors less than ε equal to zero. The goal of SVR is therefore as follows,

$$\min(0.5\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*))$$

$$subject\ to \begin{cases} y_i - (w, x_i) - b \leq \varepsilon + \xi_i \\ (w, x_i) + b - y_i \leq \varepsilon + \xi_i^* \end{cases}$$

The term C is known as the regularisation parameter and controls the trade-off between minimising errors and penalising overfitting. In our modelling we fix the parameter ε at one tenth of the interquartile range of the unemployment rate. The regularisation parameter is set similarly. Following Smola and Scholkopf (2003), Gunn (1998) and Vapnik (1995) we solve this problem in its dual form so we can extend the functional form to follow a Gaussian function.

A.6. Partial Least Squares

Partial least squares is a methodology similar to principal components in that we are trying to find several factors that can be used instead of a larger data set. Where partial least squares differs is that it takes into consideration the variable that is being forecasted as well as the explanatory variables. We then use the factors that are created from the training data set in conjunction with the explanatory variables from the nowcast period to nowcast the unemployment rate.

B. Forecast performance exercise

Each of the algorithms are trained using an expanding window estimation. In each window, the algorithms are given observations of employment transitions data and their corresponding unemployment rate. The algorithms then individually determine the parameters associated with each employment transition state, which are then combined with the employment transitions states of the nowcast period to create a nowcast for the unemployment rate. This is done for 25 estimation windows. The first window trains the algorithms on data for 1999Q3 to 2011Q4, the parameters from this estimation are then combined with the employment transitions data from 2012Q1 to create an unemployment rate nowcast for 2012Q1. The end point is then expanded one quarter and the process is repeated until we have nowcasts of the unemployment rate from 2012Q1-2018Q1. We then examine the forecast accuracy of each of the models by calculating their root mean squared error (RMSE).

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{T}}$$

Where y_t are the actual final vintage unemployment rates over the 2012Q1-2018Q1 period and $\hat{y}_t$ are the real time nowcasts from a given model over the same period. T is the total number of nowcasts over the period of interest. The RMSE is then individually computed for each of the models and for the benchmark as well. We then determine the performance of the machine learning models compared to the benchmark by calculating the RMSE ratio for model i, this is done using the following equation

$$Ratio_i = \frac{RMSE_i}{RMSE_{AR(1)}}$$

In order to determine whether there are quantitative differences between the machine learning models and the benchmarks, we calculate the Diebold-Mariano test statistic (Diebold & Mariano 1995). This test determines whether the differences of nowcasts between any two models are statistically significant.

In practice it is generally useful to conduct a forecast combination exercise. This is well established within the literature as a way of improving forecast accuracy, furthermore this approach allows the bias of different models to be offset. We conduct a simple forecast aggregation in which we take an average of the nowcasts from all the machine learning models to create one set of nowcasts. We then assess this set of nowcasts using the aforementioned evaluation strategy.