Support Vector Machines and Neural Network Models for Regression of Australia's Unemployment Rate

Biljana Simonovikj

17/02/2021

Abstract/Executive Summary

Not taking the latest COVID-19 situation into consideration, while in recent years the unemployment rate has fallen from its peak of over 11% per cent in the early 1990s, there has been a relatively consistent trend downwards and unemployment rates went as low as 4.2% in 2008. However, the great recession that occured between 2008 and 2010 caused an increase about 2% in unemployment rate which remained steady afterwards, but that period was not predicted. In this paper, the goal is to show how chosen Machine Learning technique which is Support Vector Machine with Radial Basis Kernel with caret package and Deep Learning technique which is Vanilla Neural Networks with keras may improve short-term predicitive accuracy.

As a case study, we used Australia's unemployment rate time series data starting from June 1981 until December 2020 with related variables. We evaluated model accuracy by measuring Root Mean Squared Error, (RMSE), R Squared, (R^2) and Mean Absolute Error,(MAE) for regression of Australian unemployment rate. The algorithms were trained on Australian macroeconomics variables over the period of 1981:Q2 to 2018:Q1. The results were tested using the same variables over the period from 2018:Q1 to 2020:Q4. As a result, both algorithms are able to predict economic downturns but higher accuracy was obtained by using Vanilla Neural Network keras model with 3 layers and 64 nodes. The solution of the problem produced by SVM with RMSE = 0.6381 was a little contradictory with the actual values as assessed by graphical visualisation. It overestimated the unemployment rate which resulted in a predicted decrease of the unemployment rate over the last months of 2020 despite the observed increase in actual values. On the other hand, the Neural Network with estimated RMSE of 0.4675 in the final model demonstrated extraordinary accuracy in capturing the increase of unemployment rate and overall pattern of data.

Brief Overview of the Australian Unemployment Rate

Unemployment occurs when someone is willing and able to work but does not have a paid job. It is measured as percentage of people in the labour force who are "unemployed." Statistical summaries of the labour force, such as those produced by the Australian Bureau of Statistics (ABS), Australia (2008), divide the population aged 15 and over into "employed," "unemployed" and "not in the labour force." In general, the ABS calculates Australia's unemployment rate using only the labour force, thus, the unemployment rate is an important measure for the Australian government because it provides a meaningful insight into the state of the national economy and the general well-being of citizens.

Unemployment in the Past and Today

Today, the outlook for the Australian and global economies is being driven by the COVID-19 pandemic in the past two years. The onset of COVID-19 pandemic saw unemployment climb in Australia economy from 5.4 to 7.5 % in June, 2020. Since today, the unemployment rate decreased 0.2 points to 6.4% and it was at steady level of 5.2% in the last quarter of 2019. The 5.2% is considered as steady level of unemployment rate that follows a five-year downward trend to reach peak at 6.3% in 2014, Downes, Hanslow, and Tulip (2014). Over the last 20 years, there have been two such periods when the unemployment rate has experienced a steady decline: in the early 2000s when unemployment was approximately 7% in the midst of a post-recession recovery; from 2001 until 2008, employment participation had been rising and unemployment had been falling until a low of 4.2% that was reached in 2008. However, Global Financial Crisis (GFC) in 2008-2009, marked something of a turning point for the Australian labour market and it caused the unemployment rate to increase sharply to 5.7% over the next 12 months, De Fontenay et al. (2020). The unemployment rate did jump around 2 percentage points in 2008-2009, but it then remained steady and did not return to its higher pre-2001 levels until the COVID-19 crisis.

The Australian Labour Market

The Australian labour market is quite dynamic and flexible in absorbing workers from 2008 until the COVID-19 crisis, with many people flowing into and out of employment, unemployment and the labour force each month, but it is characterized as weak-labor market. The weak labour market from 2008 to 2020 is reflected primarily in young workers finding lower-scored occupations and earning lower wage rates than earlier generations. The average duration of unemployment has increased steadily and the rate at which unemployed

people are able to find a job has slowed. At the same time, full-time employment declined and part-time employment increased among workers aged 15-34. Hence, over the past 20 years, half of the unemployed pool remain unemployed from month to month because around 23% of unemployed people transition into employment and a further 21% leave the labour market each month according to the analysis with micro-level labour market data, Carroll (2006). However, recently it has been published that there is usually a quite strong link between the unemployment rate and full-time employment, Burda and Hamermesh (2010). When looking for a good indicator of the state of the economy, it is better to look for underemployment rate as it measures persons who usually work less than 35 hours per week but prefer to work more hours, Cassidy et al. (2020). Some authors have argued that the risein part-time work implies that the unemployment rate has lost its relevance as a measure of labour market health, Biddle et al. (2020). This suggests that the unemployment rate may no longer be useful as the primary measure of the health of the labour market.

Unemployment Rate Predictors in the Real World

Some factors which are considered relevant to predict unemployment rate of the weak-labor market in Australia are GDP (gross domestic product), Government final consumption expenditure, final consumption expenditure of industry sectors, term of trade index, CPI (consumer price index), the number of job vacancies and the estimated resident population, McDonald (2020). GDP is a measure of consumption and investment which is affected by overall revenue and directly correlated to unemployment rate. Government final consumption expenditure is a measure of government spending on the needs of individuals or broader community and it is significant contributor to the overall GDP of Australia, contributing around 20% of the total GDP since 1975 (Bayar and others (2016)). Similarly, final consumption expenditure of all industry sectors considers the total spending of all Australian Industries, which is also a great contributor to the overall GDP of Australia. Terms of Trade Index is calculated as the ratio of export prices to import prices, if it increases, it is considered a favourable movement in the terms of trade. Consumer price index (CPI) is defined by the ABS, Australia (2008) as "a measure of the average change over time in the prices paid by households for a fixed basket of goods and services." It is primarily used to monitor and evaluate levels of inflation in the Australian economy. The number of job vacancies and the estimated resident population are also key factors as they give an indication of the pressures and availability of jobs within Australia.

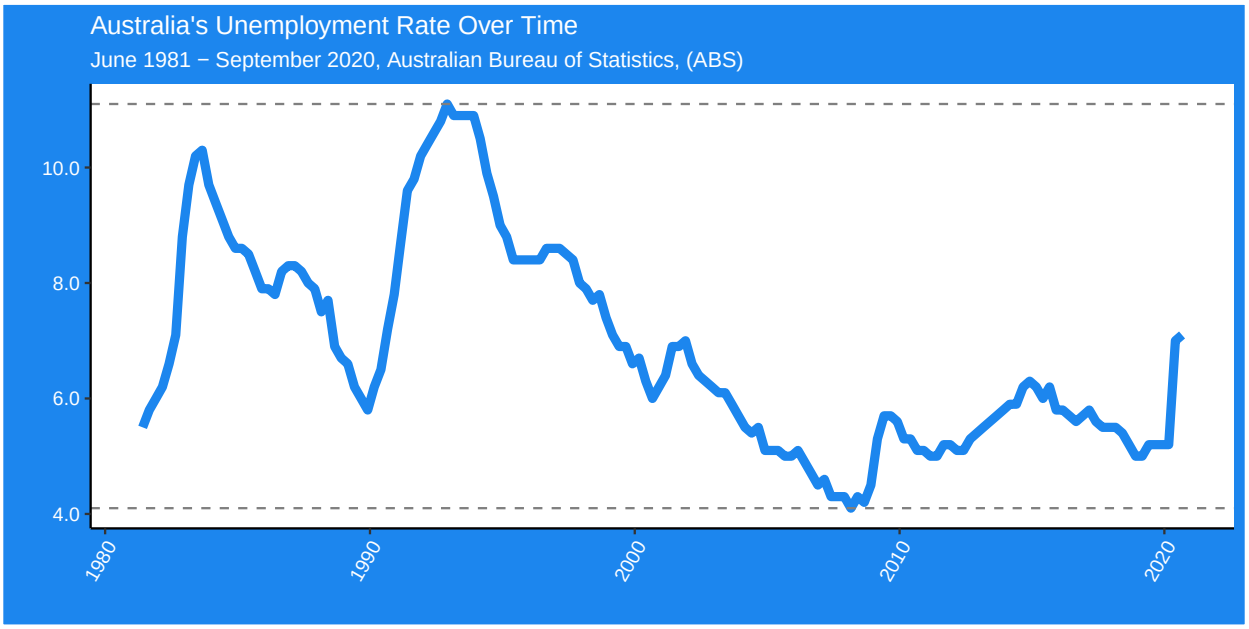


Figure 1: This statistics presents the unemployment rate in Australia from 1980 to 2020. The unemployment rate is cyclic for the first 20 years, but random for the last. It shows large decreases over economic downturns - most notably during the early 1990s and the Global Financial Crisis (GFC) between 2008-2010.

Table 1: Data dictionary for AUS Data dataset

Input	Name	Type	Description
1	Period	Date/Time continuous	Represents each three month period of the year
2	Gross domestic product: Percentage	Numeric continuous	Percentage estimates in Gross domestic product
3	General government ; Final consumption expenditure: Percentage	Numeric continuous	Percentage estimates in the Government final consumption expenditure
4	All sectors ; Final consumption expenditure: Percentage	Numeric continuous	Percentage estimates in final consumption expenditure of all industry sectors
5	Terms of trade: Index - Percentage	Numeric continuous	Term of trade index (percentage)
6	CPI (all group)	Numeric continuous	Consumer Price Index of all groups (CPI)
7	Job vacancies (000)	Numeric discrete	Number of job vacancies measured in thousands
8	Estimated Resident population(000)	Numeric discrete	Estimated Resident Population measured in thousands
Output	Name	Type	Description
1	Unemployment rate Percentage	Numeric continuous	Unemployment rate measured in percentage

Aim of the Project

The goal of this report is to work through machine learning and deep learning problem from end-to-end on a time series dataset to predict the Australia's unemployment rate. In general, this predictive modeling project can be broken down into about 6 common tasks: define the problem, summarize the data, prepare the data including sub-setting the data, evaluate supervised machine learning algorithms (Support Vector Machine with Radial Basis Kernel, SVM and Random Forest, RF) and a Vanilla Neural Network, VNN to the prepared data, analyze the predictive performance on training and testing data and compare the results.

Data

(i) The source of the data is Australian Bureau of Statistics (ABS), https://www.abs.gov.au/about?OpenDocument&ref=topBar.

(ii) The dataset of interest, called "AUS_Data.xlsx," is aggregated from monthly to quarterly time spans for the assignment purpose. The data is available quarterly from June 1981 to December 2020.

(iii) The sample size is 158 rows.

(iv) The dataset contain 7 input/predictor variables and 1 output/response variable. All variables are continuous/measure, except Period which is date/time continuous; Job vacancies (000) and Estimated Resident population(000) are discrete (Table 1).

(v) Prepearing data for analysis: a) changing column names of all variables to "date," "GDP," "GGFCE," "ASFCE," "TOT," "CPI," "job_vacancies," "ERP" and "unemployment_rate"; b) changing format of date column with lubridate package and deriving year and month from the date attribute; c) coercing data types of "job_vacancies" and "ERP" into numeric data types; d) checking missing values, data type and unique values with `df status` function from funModeling package. There were 5 NA's detected at "job_vacancies" and "ERP" variables each which were imputed by linear interpolation algorithm with imputeTS package as the best suitable type of imputation to time-series data, Moritz and Bartz-Beielstein (2017);

(vi) Pre-processing interventions: a) in order to interpret data using descriptive statistics we calculated the measures of central tendency (min, max, mean and median) and measures of dispersion (standard deviation, variance, IQR and skewness) as shown in Table 2; b) unimodal analysis (histograms, density plots, box plots) and multimodal analysis (scatter plots) of target variables including correlation analysis with `cor` function were performed; c) the dataset was partitioned without `set.seed` function into its training and testing datasets in which the train data included 147 observations before March

Table 2: Descriptive Statistics

	Max	Min	Median	Mean	SD	Variance	IQR	Skewness
unemployment_rate	11.1	4.1	6.2	6.9	1.8	3.2	2.8	0.7
GDP	3.3	-7.0	0.7	0.7	1.0	0.9	0.7	-3.0
GGFCE	7.5	-4.6	1.0	0.9	1.7	2.8	1.5	-0.1
ASFCE	5.9	-8.3	0.8	0.8	1.1	1.2	0.8	-2.8
TOT	13.2	-8.1	0.3	0.3	3.0	8.8	2.8	0.7
CPI	116.6	28.4	73.5	75.1	25.0	625.9	37.8	0.0
job_vacancies	232.3	26.8	102.5	113.1	57.3	3287.1	94.4	0.3
ERP	253643.1	149232.6	191831.1	196711.8	30659.5	940002514.7	49063.8	0.3

2018 whilst the test dataset consisted of 11 observations form March 2018 to December 2020. The date variable was removed from training and testing datasets. The training and test unemployment rate columns were saved in respective vectors. The results from the table shows that GDP and ASFCE have skewness of -3.0 and -2.84 respectively, or skewed distribution to the left which is caused by outliers. No outliers were omitted from the dataset in order to maintain data integrity. Except, outliers were replaced with median values from GDP, ASFCE and GGFCE variables. Standard deviation is a measure used to quantify the amount of variation of a set of data values from its mean. For instance, since the mean of unemployment_rate is 6.86 and its standard deviation is 1.79, we can estimate that approximately 95% of the values will fall in the range of $6.86-(2*1.79)$ to $6.86+(2*1.79)$ or between 3.28 and 10.44.

(vii) Data normalization: we standardize the data with z-score normalization by columns of predictor variables, where each column was rescaled to have zero mean and standard deviation 1, $xnorm = (x-\mu)/\sigma$ with the `scale` function. Mean and standard deviation values of training dataset were used for rescaling testing dataset. The goal of normalization which is very important to VNN is to change the values of numeric columns in the dataset to a common scale with aim to:

- speed up learning process which will lead to faster convergence;
- ensure a feature has both positive and negative values which makes it easier for the weight vectors to change directions. This helps in making the learning flexible and faster by reducing the number of epochs required to reach the minima of the loss function;
- ensure that the magnitude of the values a feature assumes fall within a similar range. The network regards all input features to a similar extent, irrespective of the magnitude of the values they hold.

Developing machine learning algorithms was performed with caret package, whereas developing the network architecture for a VNN was performed with kears and tensorFlor packages. Both methods were implemented in R version (4.0.3).

(viii) only for keras protocol, after normalization step, additional step was introduced, because kears requires all predictors and response variables to be numeric in a matrix form which was accomplished with `as.matrix` function after data partitioning. The training and test unemployment rate columns saved as vectors were converted into matrices as well.

Analysis and Investigation of Machine Learning (ML) method

We select two algorithms capable of working on this regression problem implemented with caret package. The 2 algorithms selected include:

- non-linear algorithm: Support Vector Machines (SVM) with a radial basis function
- Random Forest, (RF) with bagging technique for decision trees

The data is small-sized, times series data, nonlinear and non-stationary because it contains trends and seasonal effects shown quarterly. SVM is a supervised model that can solve linear or nonlinear problems, especially on small datasets. This algorithm tries to create a decision boundary that has maximum margin

and optimal hyperplane. The boundaries of data can be obtained by using the convex hull, Yang, Chan, and King (2002). The key formation of SVM's is a kernel function, that uses convex hull to choose the extreme points. SVM for regression is considered a nonparametric technique because it relies on kernel functions. The output model from SVR does not depend on distributions of the underlying dependent and independent variables, instead it depends on kernel functions, Trafalis and Ince (2000). The commonly used kernel functions are: a) Linear, b) Polynomial, c) Sigmoid and d) Radial Basis. Given a non-linear relation between the variables of interest and difficulty in kernel selection, we select Radius Basis Function (RBF) kernel as the default kernel. The kernel function transforms our data from non-linear space into a higher dimensional feature space where it is easier to find a separating hyperplane and where a linear threshold can be used, Wauters and Vanhoucke (2014). The kernel trick allows the SVR to find a fit and then data is mapped to the original space. The non-linear SVM results will be a linear combination, but with new variables, which are going to be derived through a kernel transformation of the prior predictors' ratios.

The advantages of the SVM technique can be summarised as follows: a) SVM with Radial Kernel has a flexibility in the choice of the form of the threshold that separates the unemployment rate from less important predictors, and no assumptions about the functional form of the transformation is necessary. b) if the hyper parameters C and sigma are appropriately chosen, SVMs can be robust, even when the training sample has some bias, Awad and Khanna (2015); c) SVMs deliver always a unique solution, since the optimazation problem is convex. This is an advantage compared to VNNs, which have multiple solutions associated with local minima and for this reason may not be robust over different samples.

Random Forest is very robust algorithm also used both in classification and regression and is based on the bagging and ensemble learning technique. It creates many trees and combines the output of all the trees. In this way it reduces overfitting and the variance and therefore improves the accuracy, Lingjun et al. (2018). Most important advantages are: a) works well with both categorical and continuous variables; b) very tolerant to missing values; c) no feature scaling (standardization or normalization) required as it uses rule based approach instead of distance calculation; d) can handle successfully non-linearity and outliers; e) less impacted by noise. The main disadvanatges are: a) it can be complex by creating a lot of trees which requires more computational power and resources; b) longer training period because it generates a lot of trees and makes decision on the majority of votes.

In general, regression models can't predict beyond the range in the training data and it is known that they underestimate high values and overestimate low values in the data, Horning (2013).

Parameters and Hyper-Parameters

The streamline of model building and evaluation process involves using the `train()` function to:

- evaluate the effect of model tuning parameters on performance by using resampling method
- choose the "optimal" model across these parameters
- estimate model performance from a training dataset

`train()` function arguments applied in the models:

1. `Method` - string specifies which regression model to apply: "svmRadial" or "rf."
2. `TuneGrid` - a data frame with possible tuning values. The columns are named the same as the tuning parameters. We specify the values of sigma = seq(0.01, 1, 0.1), C = seq(1, 10, by=1) for SVM model and mtry = c(1:9) for RF model that we want to analyze with `expand.grid()` function.
3. `Metric` - a string that specifies what summary metric will be used to select the optimal model evaluation. We set to "RMSE" for this regression task.
4. `TuneControl` - a list of values that define cross-validation of the model.

The SVM model defined by caret package has two hyperparameters: "sigma" and "C":

1. `sigma` – controls the influence of individual training instances. A value too low can be to restrictive whereas a value to high can lead to overfitting.

2. C (cost) – is the regularization parameter of the support vectors in all dimensions of the hyperplane that controls training errors and margins. The cost value represents the number of mis-classifications that are allowed and thus makes the model less sensitive to outliers and reduces the risk of over fitting the training data. A large value of Cost parameter indicates poor accuracy but low value confirms bias and vice-versa. The bias variance trade off is always an important consideration in any ML model.

The Random Forest model defined by caret package has only one hyperparameter: "mtry":

1. mtry - defines the number of variables randomly sampled as candidates at each split. We specify that mtry = c(1:9) which is the number of predictors in the training dataset.

When we are building a predictive model, we need to evaluate the capability of the model on unseen data. This is typically done by estimating accuracy using data that was not used to train the model. We evaluate model accuracy using repeated 5-fold cross validation with 3 repeats. Cross validation is a gold standard for evaluating model accuracy to balance overfitting the training data. The parameters for cross-validation are assigned as part of trainControl() function.

trainControl() function arguments applied:

1. Method - determines the type of sampling/validation to be undertaken. Repeated cross validation, "repeatedcv" is chosen to estimate the tuning parameters.
2. Number - number of folds for cross validation. This is set at 5 for 5-fold cross validation.
3. Repeats - the number of repetitions of cross validation to be undertaken. This is set to 3 due to computational time.

Model Evaulation Metrics

When we use caret to evaluate the created models, the default metrics used are Root Mean Squared Error, RMSE, R Squared, (R^2) and Mean Absolute Error, MAE for regression. RMSE is the average deviation of the predictions from the observations. It is useful to get a gross idea of how well (or not) an algorithm is doing, in the units of the response variable. Values closest to zero are the best. R^2 or also called the coefficient of determination provides a goodness-of-fit measure for the predictions to the observations. This is a value between 0 and 1 for no-fit and perfect fit respectively. MAE is also a useful as it measures the average magnitude of the errors in a set of predictions, without considering their direction. Same as RMSE, values closest to zero are the best.

Models Performances and Interpretations on Training Dataset

We evaluate baseline models of SVM and RF on train data with 5-fold cross validation (each fold is 147 observations for training and 11 for test) with 3 repeats. As to SVM, tuning parameter 'sigma' was held constant at a value of 0.08232. RMSE was used to select the optimal model using the smallest value of 0.0.9541201 (Table 3). The final values used for the model were sigma = 0.9541201 and C = 1. RF performed better than SVM with smallest RMSE value of 4500681 and mtry = 5. We can also see that RF has the best fit for the data in his R^2 measures (Table 4). The training error rate is presented as standard deviations of each parameter (SESD, RsquaredSD, MAESD).

Table 3: Results of estimated SVM model accuracy on train data

sigma	C	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
0.082324	1	0.9541201	0.7240316	0.6868329	0.1401703	0.093355	0.1192678

Table 4: Results of estimated RF model accuracy on train data

mtry	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
5	0.4500681	0.9392699	0.3131367	0.1203184	0.0415745	0.0855114

We improve the accuracy of the well performing algorithms by tuning their parameters. We can see (Table 5) that the sigma values flatten out with larger C cost constraints. It looks like SVM might do well with a sigma of 0.1 and a C of 2 which gives us a respectable RMSE of 0.949117. On the other hand, tuning RF (Table 6) involves tuning only one parameter, as a result the grid search is a linear search through a vector of predictor values. We can see that the most accurate value for mtry = 5 with an accuracy of RMSE = 0.4555232. The RF algorithm created the most accurate model and we can get this idea simply by observing the plot below of predicted values for unemployment rate versus actual values.

Table 5: Results of tuned SVM model accuracy on train data

sigma	C	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
0.1	2	0.9491171	0.7300289	0.6781637	0.1228478	0.0741458	0.1098288

Table 6: Results of tuned RF model accuracy on train data

mtry	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
5	0.4555232	0.9374352	0.3150029	0.1235388	0.043237	0.0879723

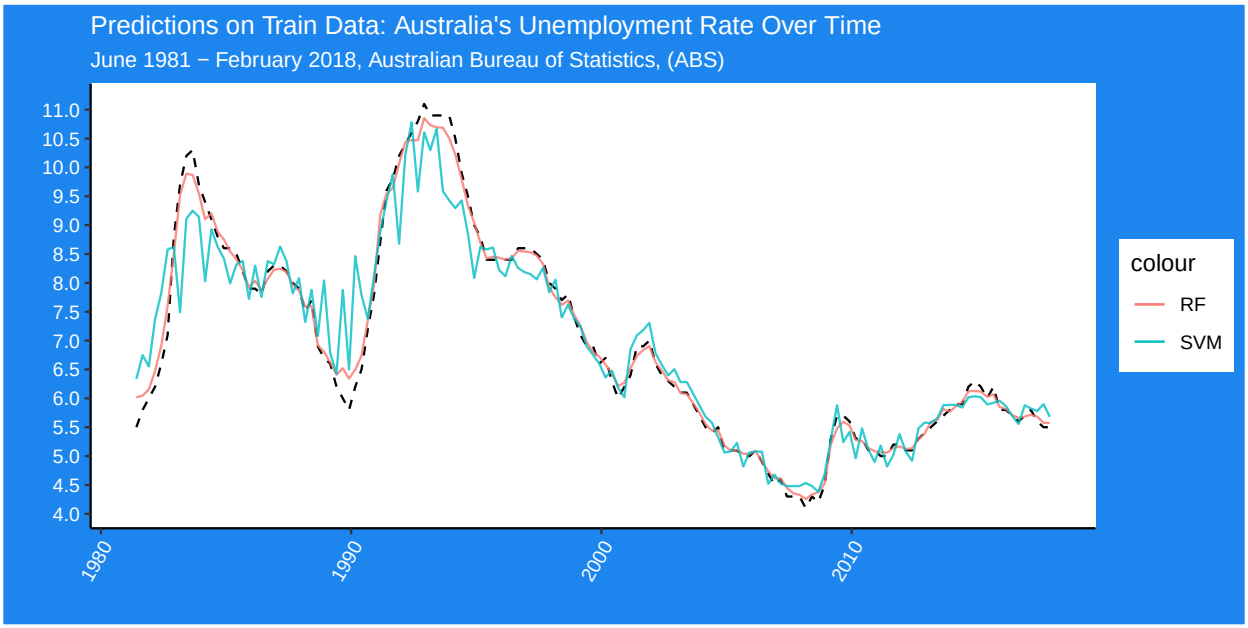


Figure 2: Predicted vs actual values of unemployment rate for RF & SVM models on train data.

Predicitive Performances of the Models on Test Dataset

Predictive performance of SVM and RF on training and test data is shown in (Table 7). Model evaluation metrics were coded as user-define functions in R. These functions are used to compare the generated train and test predictions with the actual test unemployment rate in order to determine the best predictive regression

model. Surprisingly, high values of RMSE for RF model on unseen data clearly indicate that the RF model is subject to overfitting the data. The classifier has completely mimicked the training data patterns when we are testing it on unseen data, its unable to find that patterns and returns accurate predictions. In a random forest, it happens when we use a larger number of trees than necessary. We have just eleven observations in our test dataset, then each of the samples taken with replacement from this data would consist of not more than the eleven distinct values. Shuffling the cases and not drawing some of them would not change much about its ability to learn anything new about the underlying distribution. So a small sample is a problem for bootstrap.

In addition, results from the (Table 7) indicate that the predicted R Squared for RM on unseen data is 0.0433790 or 0.4% which is very low in comparison to training data which is 99% for our model. We have reason to believe that the RF model do not predict new observations as well as it fits the training data. On the other hand, predicted R Squared for SVM model of 22% sounds better but yet, if we observe the visualization, (Fig.3) in details we can notice that from March, 2020 until September, 2020, the predicted values provided by SVM solution are decreasing instead of observed increase in actual values.

Table 7: Comparison of model evaluation metrics

	Train Data			Test Data		
	RMSE	RSquare	MAE	RMSE	RSquare	MAE
RF	0.1586960	0.9922078	0.0159564	0.7069758	0.0433790	0.0994519
SVM	0.5570141	0.9040027	0.0490789	0.6381833	0.2204899	0.1063324

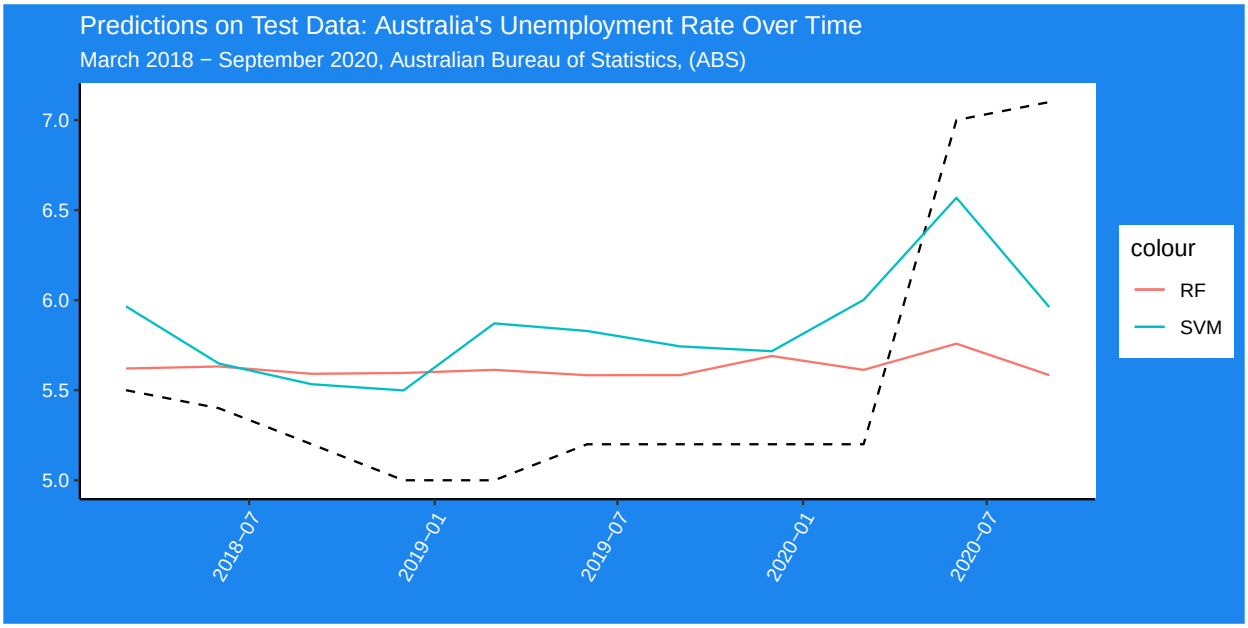


Figure 3: Predicted vs actual values of unemployment rate for RF & SVM models on unseen data.

Neural Network

Artificial Neural Network (ANN) was inspired by neurophysiologist and mathematician McHullock & Pitts, McCulloch and Pitts (1943) that described the functioning of neurons in the human brain. The first ANN was developed in an attempt to help computers simulate the way a human brain works, using a network of neurons to process information. The simplest type of ANN is called Single-Layer Perceptron or "Vanilla" Neural Net, (VNN) sometimes called the single hidden layer back-propagation network. Besides Single-Layer

Perceptron, there is a Multilayer-Perceptron (MLP) or feedforward DNN. In fact, general architecture of ANNs consists of three major layers:

- Input Layer receives the inputs from the training dataset and pass on the information to the hidden nodes. No computation is performed in any of the input nodes.
- Hidden Layers follows the input layer. There can be one or more hidden layers. It facilitates forward and backward passes and also helps in minimizing the error with each pass. The hidden nodes perform computations and transfer information from the input nodes to the output nodes. A collection of hidden nodes forms a "Hidden Layer."
- Output Layer performs computations, generates predictions and transfers the information to the outside world.

In order to develop the network architecture for a VNN, first of all we need to install kears and tensorFlor libraries in R. Due to the data transformation process that VNN is performing, they are highly sensitive to the individual scale of the predictor values. Consequently, at the pre-processing step we normalized the predictors with z-score normalization and applied the additional step for matrix transformation on train and test datasets including train and test labels as described in Data section. In addition, VNN requires all predictor variables and response to be numeric. Consequently, the date column was removed from the dataset, the same as we did for Machine Learning algorithms.

Next important steps to consider when developing a network architecture are: a) layers and nodes and b) activation.

Layers and Nodes

The layers and nodes are the building blocks of VNN which define the model's capacity. The number of nodes in a layer is referred to as the network's width while the number of layers in a model is referred to as its depth. Layers are considered dense (fully connected) when all the nodes in each successive layer are connected. Therefore, the more layers and nodes we add the more opportunities for new features to be learned. We initiated our sequential VNN architecture with `(keras model sequental)` function and then added three dense layers with `layer dense` function. The first input dense layer contains the predictors which were passed as the number of predictor columns to `input shape` argument of the function. Nevertheless, the successive layers are able to dynamically interpret the number of expected inputs based on the previous layer.

Hidden Layers

There is no well-defined approach for selecting the number of hidden layers and nodes. Hence, the aim is to find the simplest model with the most efficient performance. Usually with regular tabular data, 2–5 hidden layers are often sufficient. The number of nodes is largely determined by the number of features in the data but it is not hard requirement. Hastie, Tibshirani, and Friedman (2017) stated that a typical network contains between 5 to 100 neurons per layer in which it is better to have more than less as lighter networks tend to have a reduce flexibility with complex dataset. We have to bear in mind that when dealing with many features in the data and, therefore, many nodes, training deep models with many hidden layers can be computationally more efficient than training a single layer network with the same number of high volume nodes, Goodfellow et al. (2016). The dimension of our training dataset is 9 variables (7 predictors plus two variables year and month derived from date variable), therefore, the input layer and the three hidden layers contained 64 units (neurons). The choice of output layer is driven by the modeling task. For regression problems, our output layer contained one node that outputs the final predicted value.

Activation

A key component with neural networks is activation which refers to the same activation process of neuron activation in the human brain. Each node is connected to all the nodes in the previous layer. Each connection gets a weight and then that node adds all the incoming inputs multiplied by its corresponding connection weight plus an extra bias parameter. The summed total of these inputs become an input to an activation function. It is a mathematical function that determines whether or not there is enough informative input at a node to fire a signal to the next layer. There are several activation functions: linear, sigmoid, softmax,

rectified linear unit(Relu), TanH, Swish, the Leaky Relu and etc. As we have rectangular data, the most common approach is to use ReLU activation functions which is simply taking the summed weighted inputs and transforming them to a 0 (not fire) or >0 (fire). For the output layers we dont specify any activation function. To reduce the bias, `bias regularizer` parameter was set to regularizer_l2(0.01) in order to apply penalty on each layer bias during optimization, Gulli and Pal (2017). .

Backpropagation

Forward pass is the first run when VNN will select a batch of observations and fed into the network with randomly assign weights across all the node connections to generate the initial predictions. Then, the error function is computed by checking how far away the prediction is from the known true value. There are many algorithms that can minimize the error function but a training algorithm is needed that is very computationally efficient and that is backpropagation algorithm with gradient descent. This algorithm calculates how much the output values are affected by each of the weights in the model. The result of backpropagation is a set of weights that minimize the error function. Running the entire training dataset through the backpropagation process is called an epoch. Typically, a batch of samples is run in one big forward pass, and then backpropagation is performed on the aggregate result. The batch size and number of batches used in training, called iterations, are important hyperparameters that are tuned to get the best results. Hyperparameters related to the neural network structure are: number of hidden layers, droput, activation function, weights intialization. Hyperparameters related to the training algorithm are: learning rate, momentum, optimizer algorithm, epoch, iterations and batch size.

To perform backpropagation we need two things: an objective (loss) function and an optimizer. For this regression problem, the loss function is mean square error (MSE). On each forward pass the VNN will measure its performance based on the loss function chosen. The default optimizer for regression is RMSProp which is tunable hyperparameter. Since choosing the best model is primarily based on trial and error, we performed many experiments with RMSProp optimizer and Adam optimizer. We achieved better results with Adam optimizer, an algorithm for first-order gradient-based optimization of stochastic objective functions and, with hyperparameter tuning applied as follows: lr = 0.001, beta_1 = 0.9, beta_2 = 0.999. To incorporate the backpropagation in our code sequence we use `compile` function that has loss, optimizer and metrics argument specified. We use mean absolute error as metric argument.

Model Training

It is performed with `fit` function that has the following arguments applied: x and y for predictor and response variable matrices, batch_size (values are typically provided as a power of two that fit nicely into the memory requirements of the GPU or CPU hardware like 32, 64, 128, 256, and so on until several hundreds), epochs (the number of times the algorithm sees the entire data set, thus, the more complex the dataset, the more epochs we'll require for our model to learn), verbose (set to FALSE but if TRUE we will see a live update of the loss function). Plotting the output shows how our loss function (and specified metrics) improves for each epoch.

K folds cross validation (CV) was used, k = 4 and epoch = 300, to determine the optimal epoch level for the final model. Set.seed was initialised using the `set.seed` function in tensorflow library, but it seem that it did not have any effect because each run resulted in different values of parameters. The optimum VNN was saved with `save model tf` and loaded into R using `load model tf`.

For this task, more than 30 experiments were performed, but only 4 models were included in the analysis. The selected 4 models have the following properties:

- Model 1 - 3 hidden layers, 64 neurons, bias regularizer and Adam optimizer
- Model 2 - 3 hidden layers, 32 nodes, bias regularizer and Adam optimizer
- Model 3 - 4 hidden layers, 32 nodes, bias regularizer and Adam optimizer
- Model 4 - 4 hidden layers, 128 nodes,bias regularize and Adam optimizer

There is no magical formula to determine the right number of layers or the right size for each layer. We evaluated an array of different architectures (on the test data) in order to find the correct model size for the data. The general workflow discovered that 3 layers network while keeping our parameters the same showed

Table 8: Comparison of model evaluation metrics

	Hidden Layer	Node	RMSE	RSquare	MAE
Model 1	3	64	0.4675584	0.5815895	0.0512237
Model 2	3	32	0.6506521	0.1897323	0.0943928
Model 3	4	32	0.5553440	0.4097240	0.0615950
Model 4	4	128	1.0002610	0.1483297	0.0615953

diminishing returns with regard to validation loss. The bigger network with 4 layers started to overfitting almost immediately, after just 20 epoch, and it overfits much more severely.

The (Table 8) shows that increasing the the number of hidden layers from three to four substantially decreased the accuracy of VNN which was easily intrpreted by the plots of predicted versus actual values on test data. However, despite large RMSE, the MAE is much lower than the other two metrics. This is because RMSE is very sensitive to large prediction errors unlike MAE which is why most investigations use this metric as the main one to determine the overall success of regression models.

The amount of neurons per layer was also tested to see the predictive power other VNN. The lighter network with 32 and 64 nodes per layer did a better job in predicting the response variable according to the obtained metrics in (Table 8). It was mention previously, that a lighter VNN with similar parameters to the optimal model would yield similar results because of the small amount of features and observations in the dataset. Our analysis confirmed this statement as correct because the evaluation metrics at network with more nodes did not show diminishing returns regarding validation loss.

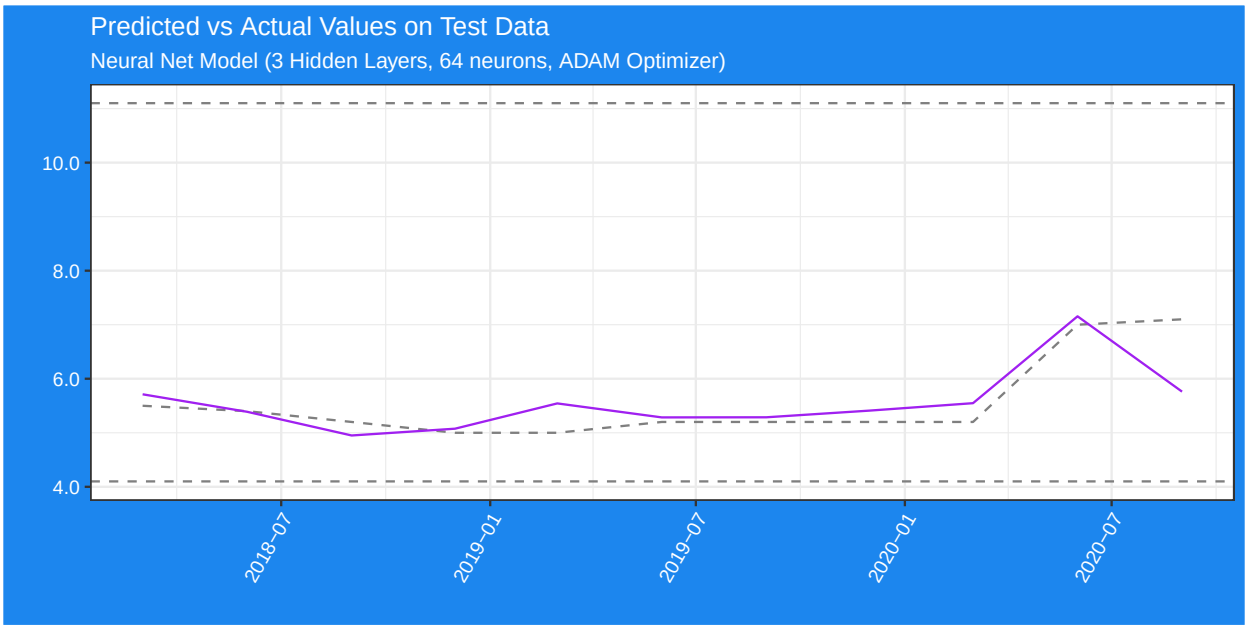


Figure 4: Predicted vs actual values of unemployment rate for VNN model on unseen data

Comparison and Suggestions

Our analysis confirmed that VNN network with 3 hidden layers and 64 nodes has a solution and predicitve ability to predict the unemployment rate with RMSE = 0.4675584 and R^2 =0.5815895 which means that the model fitted 58% of the data. The SVM with Radial Basis also has predicitve powers to predict the unemployment rate on unseen data with RMSE = 0.6381833 and R^2 = 0.2204899 but a little less efficiently than a VNN.

Obtaining the best possible results on this task can only be done with large ensembles of models. Ensembling via a well-optimized weighted average is usually good enough. We know that diversity is strength. The best

ensembles are sets of models that are as dissimilar as possible (while having as much predictive power as possible, naturally). Recommended model on this task would be ensemble of NN, SVM, XGBoost, GBM, Cubist and many other powerful algorithms.

Cross-Validated Predictive Accuracy

Accuracy of both models was assessed after performing cross validation, 5-fold with 3 repeats for SVM and 4-fold for VNN with keras according to the straightforward code sequence provided in the notes of JCU. To evaluate and compare our network with SVM performance accuracy, we split the data into training and testing datasets. The testing dataset contains only 11 observations and has a high variance which would prevent us from reliably evaluating the models. Consequently, the outliers of all data points were replaced with median values in the pre-processing step of the analysis. Hence, the best practice in such situations is to use K-fold cross-validation. The validation score for the model used is then the average of the K validation scores obtained.

Computational Training Time

The computational training time between the two models varies less than expected given the training dataset size. Computation time of SVM was just 0.222 minutes to do one run of 5-fold cross validation training over the training dataset. In contrast the VNN took from 1.1 to 1.2 minutes to do one cross-validated run. This is 0.40 times longer which is neglectable. Although not a significant issue on a small dataset this difference in training time can quickly expand for larger datasets.

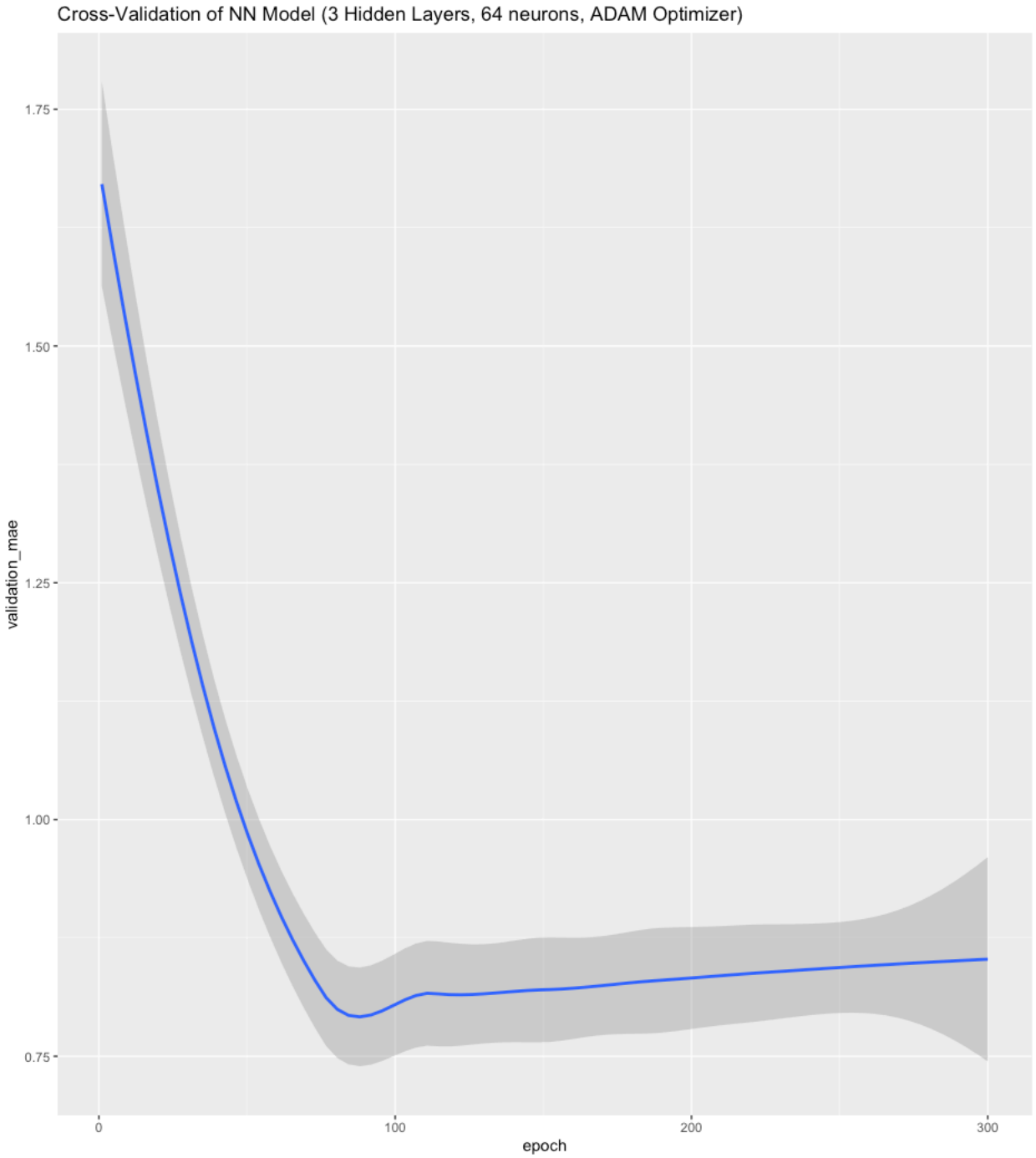
A general way of looking at the efficiency of embedding a problem in a VNN comes from a computational complexity point of view. Time complexity of VNN is related to the certain number of steps it has to undertake, memory size (space complexity), and length of algorithm (Kolmogorov complexity), Abu-Mostafa (1986). In a VNN simulation the number of computations is a measure of time complexity, the number of units is a measure of space complexity, and the number of weights (degrees of freedom) is a measure of Kolmogorov complexity. When we are trying to solve a problem with a VNN, we seek to minimize, simultaneously, the resulting time, space, and Kolmogorov complexities of the network. If a given problem is very demanding in terms of space complexity, then the required network size is large and thus the number of weights is large, even if the Kolmogorov complexity of the algorithm is very modest. VNN solutions of problems with short algorithms and high-space complexity are very inefficient. But on the other hand, problems which require a very long algorithm such as pattern recognition, then VNNs are the most efficient becuase the capacity of the net grows faster than the number of units.

Interpretability

The algorithms like SVM with Radial Basis Kernel and VNNs are very complex algorithms and very accurate for predicting nonlinear and rare phenomena. Unfortunately, more accuracy often comes at the expense of interpretability, they are difficult to interpret which is crucial for business documentation, regulatory oversight, and human acceptance and trust. Usually, these models are considered "as black boxes" due to their inner-complexity. There are multiple packages developed in the past years that provide robust machine learning interpretation capabilities which are completely based on model agnostic-approaches for interpreting the models such as: LIME, DALEX and etc. We have studied only model specific approach for interpreting the models which is based on understanding only feature importance. In model-agnostic approaches, the model is treated as a "black box." The separation of interpretability from the specific model allows us to easily compare feature importance across different models.

If a greater number of predictor variables and observations were implemented into the model, this may have improve prediction power of the VNN. On the other hand, it is also possible that additional macroeconomics and microeconomics variables of unemployment are necessary to be included into analysis. We have to note that this is time series data which need to be analyzed as time series data because our analysis without the date variable disregarded each quarter which affected the forecasting performance of the VNN. Various methods as part of advanced deep learning for times series data should be investigated further such as Recurrent Neural Networks (RNNs) that could provide better analytics of unemployment rate in Australia.

Apendices


Cross-Validation of NN Model (3 Hidden Layers, 64 neurons, ADAM Optimizer)
validation_mae
1.75
1.50
1.25
1.00
0.75
0
100
200
300
epoch

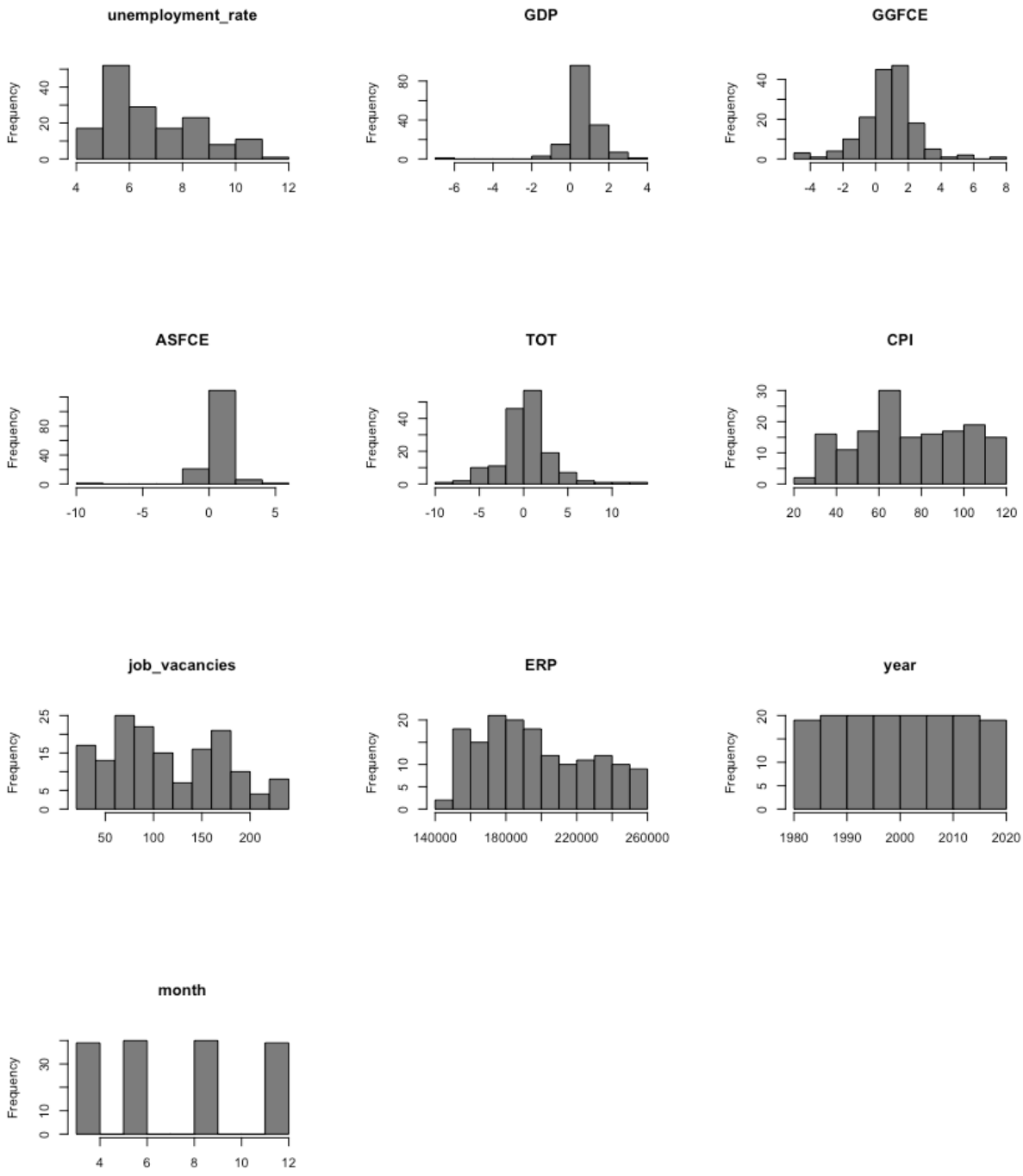


Figure 1: Univariate analysis: Histograms

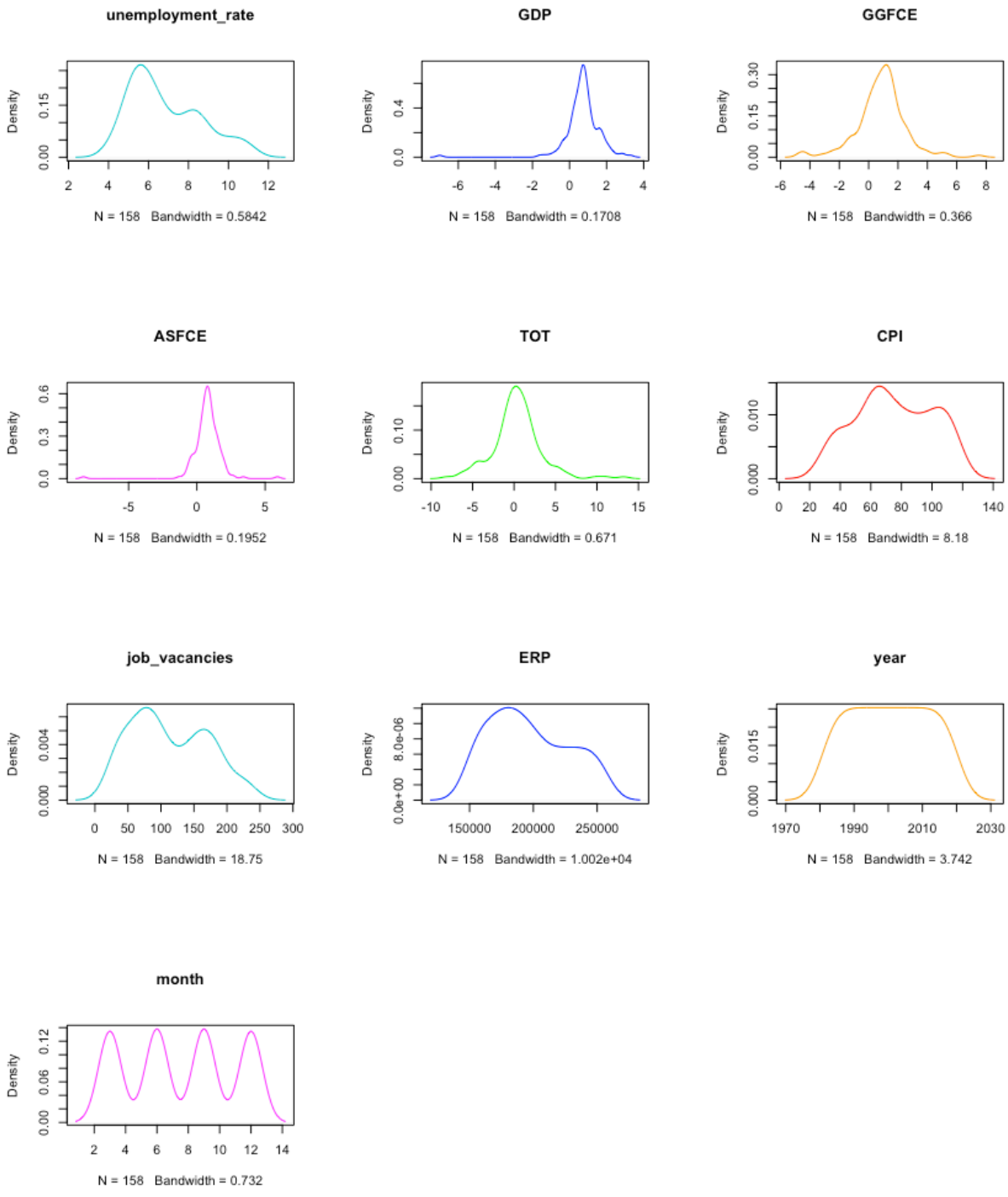


Figure 2: Univariate analysis: Density plots

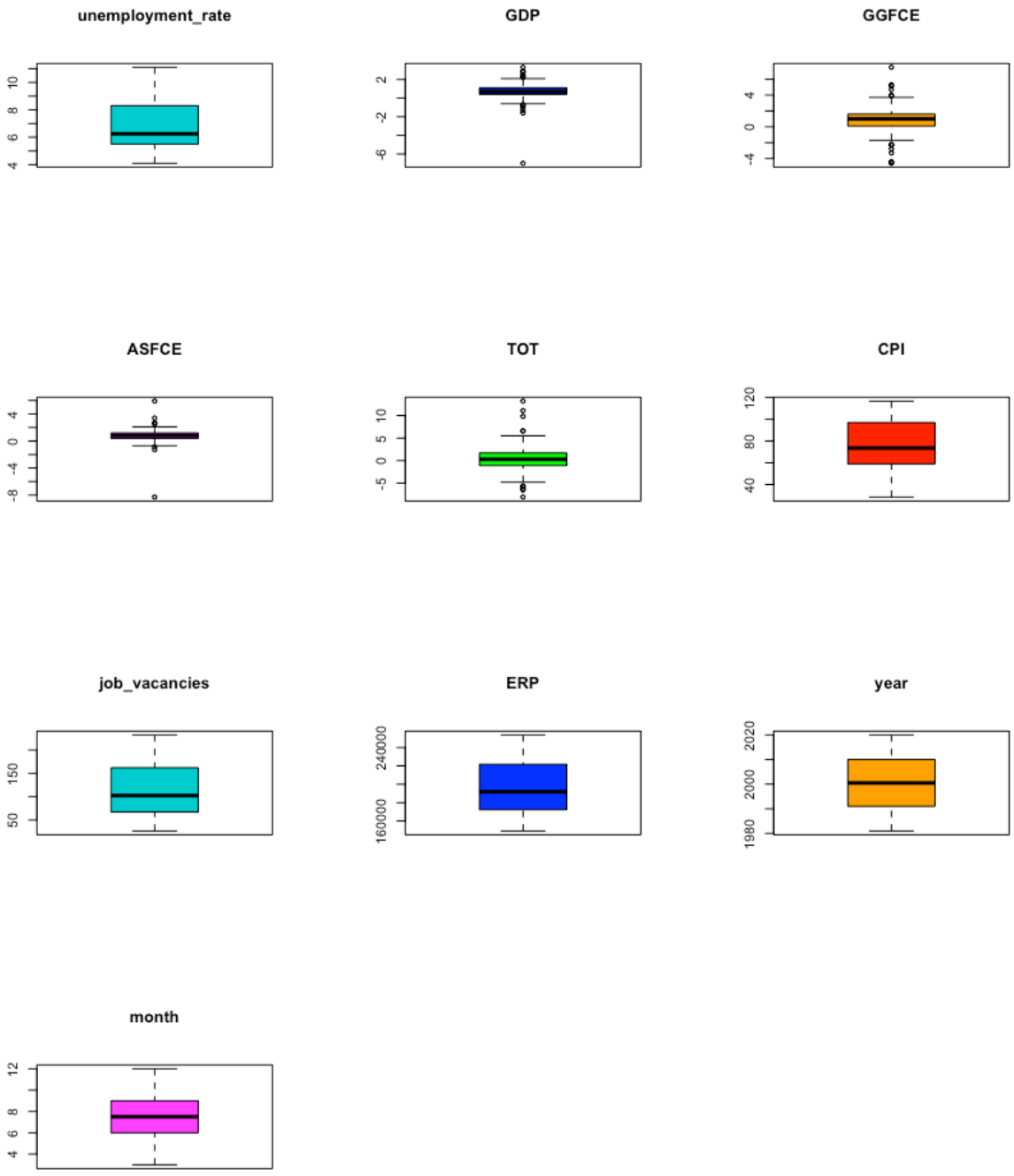


Figure 3: Univariate analysis: Box plots

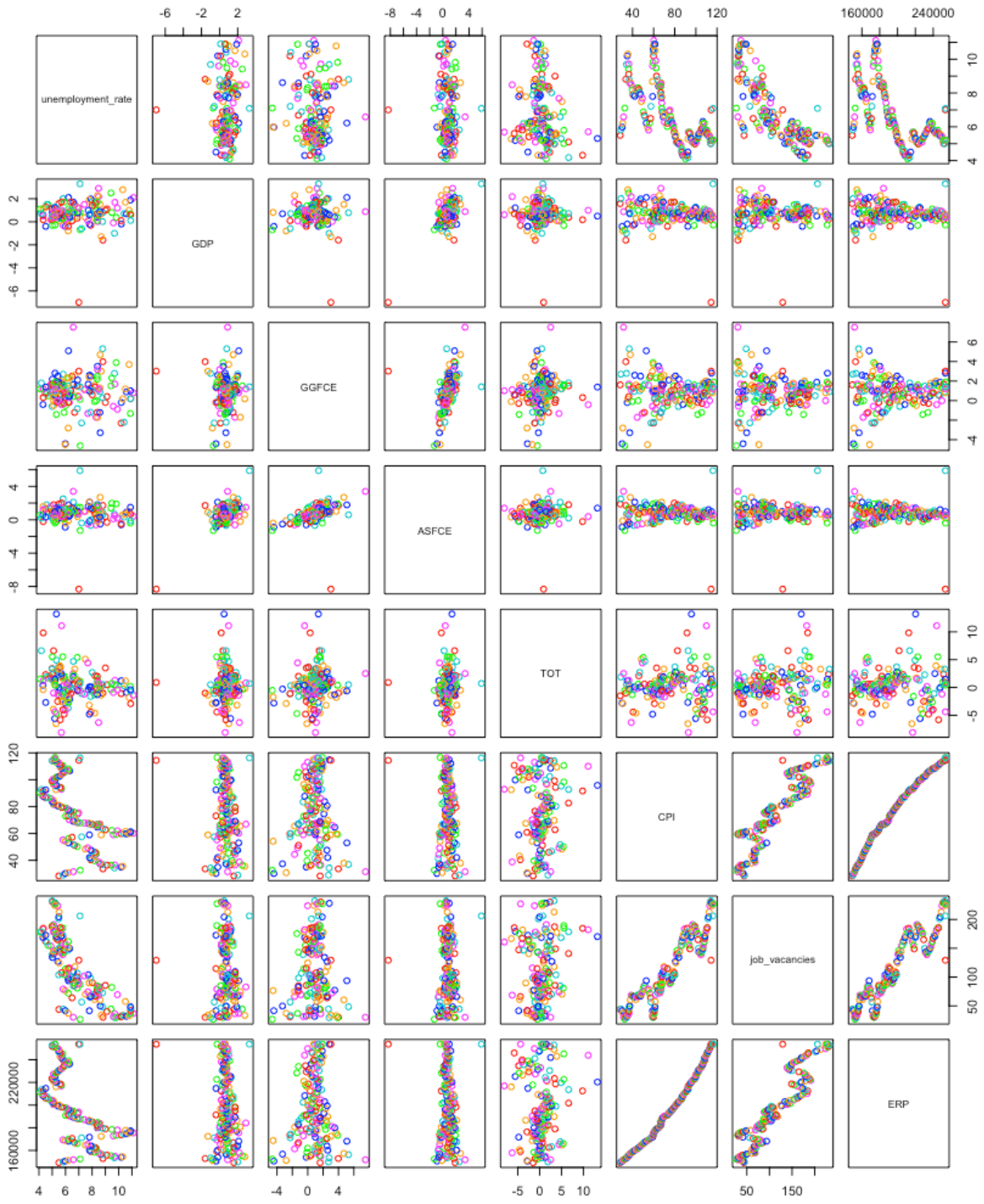


Figure 4: Multivariate analysis: Scatter plots

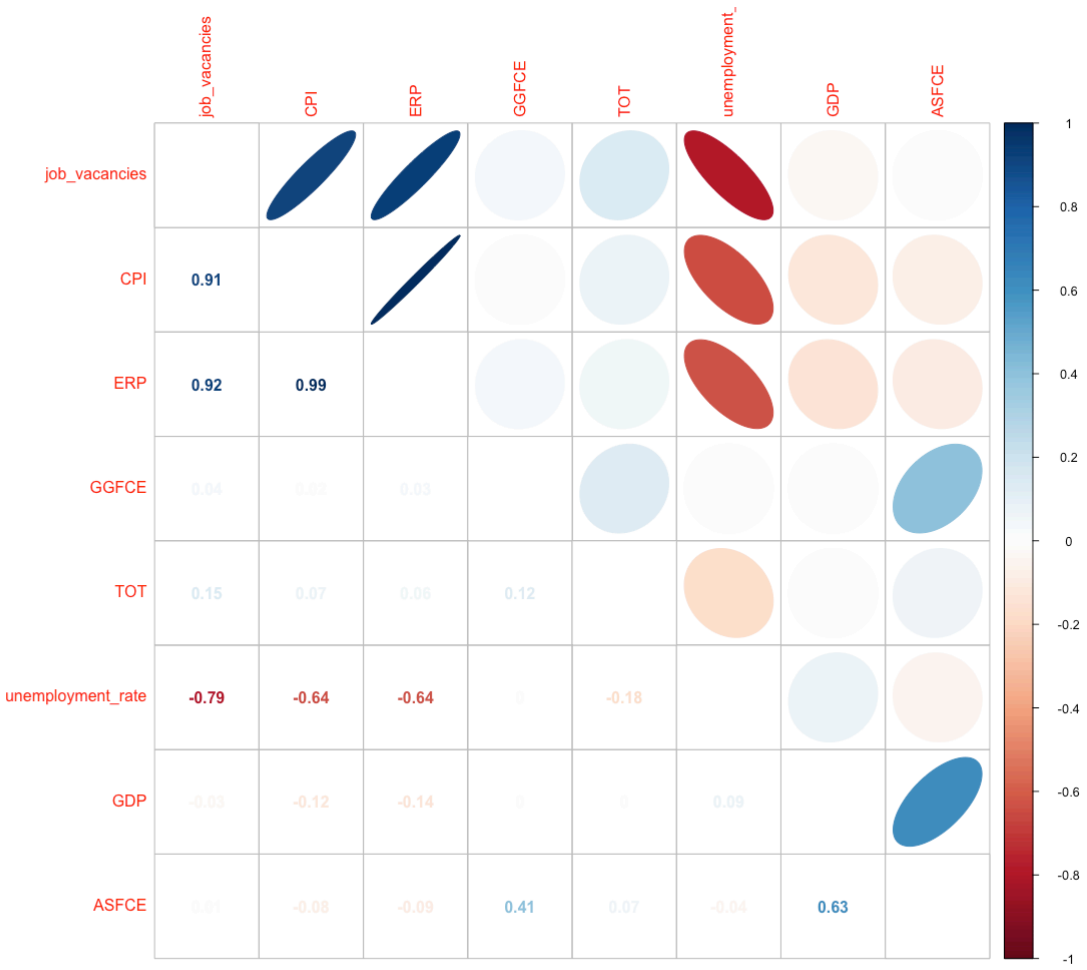
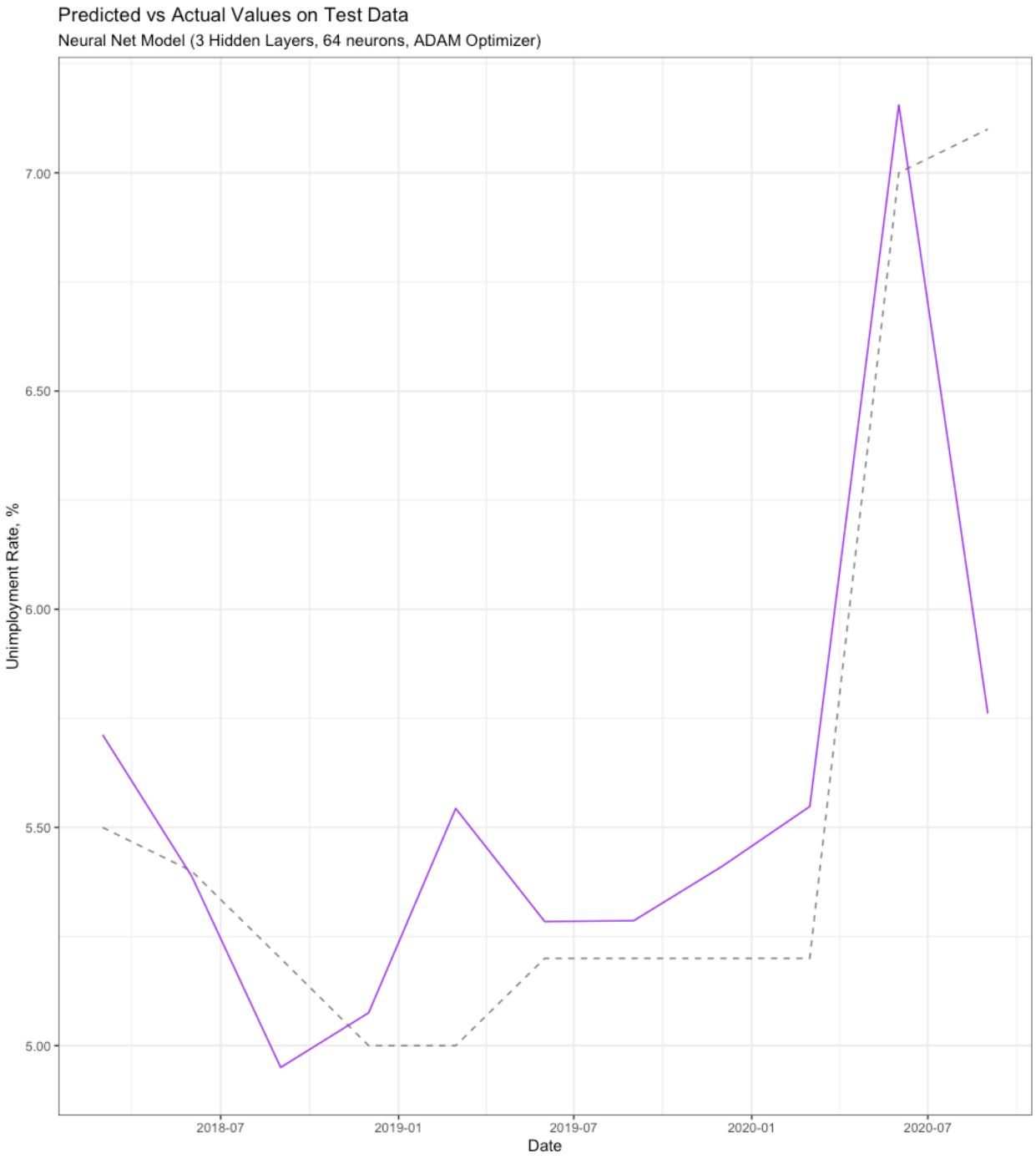
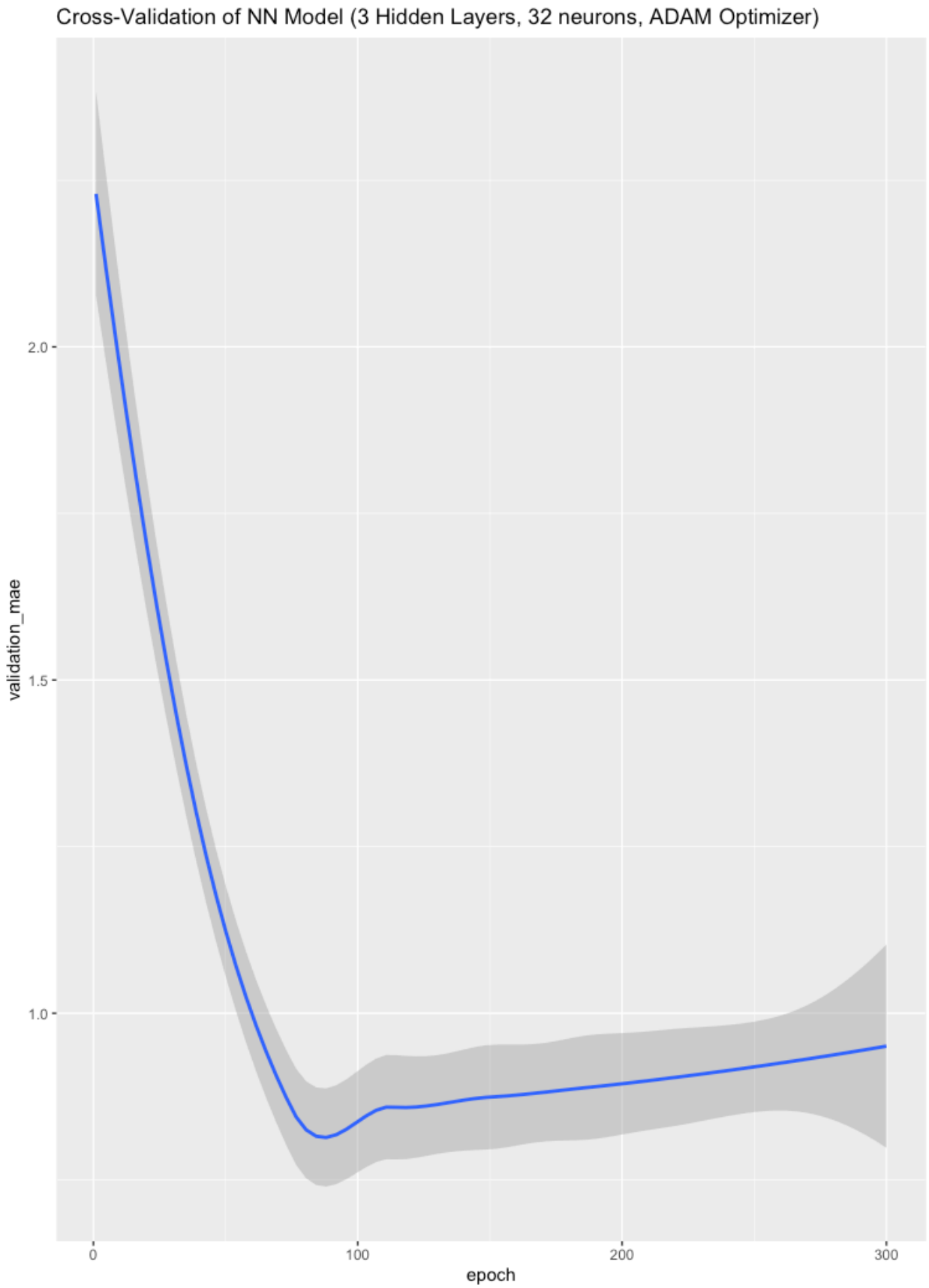
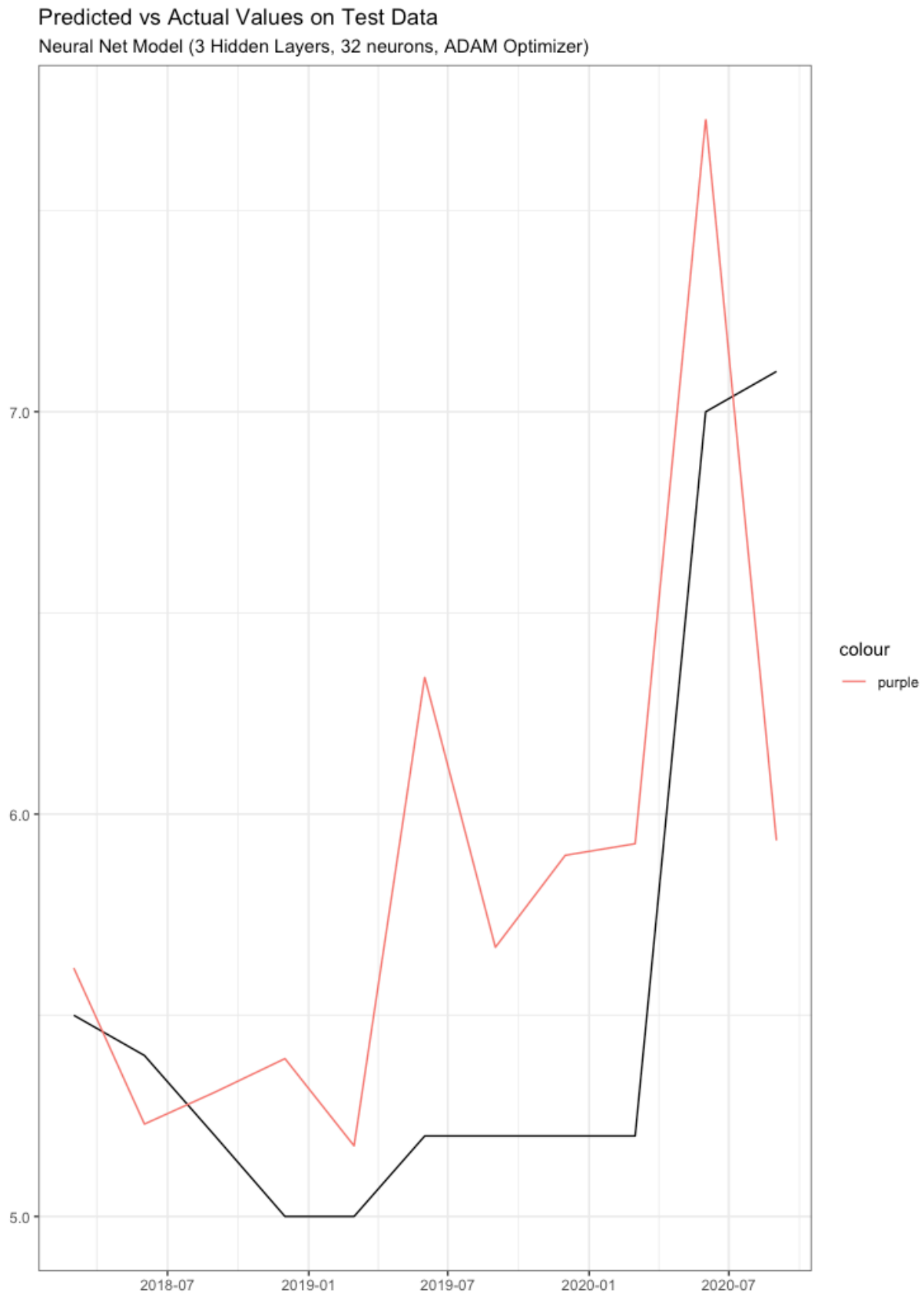
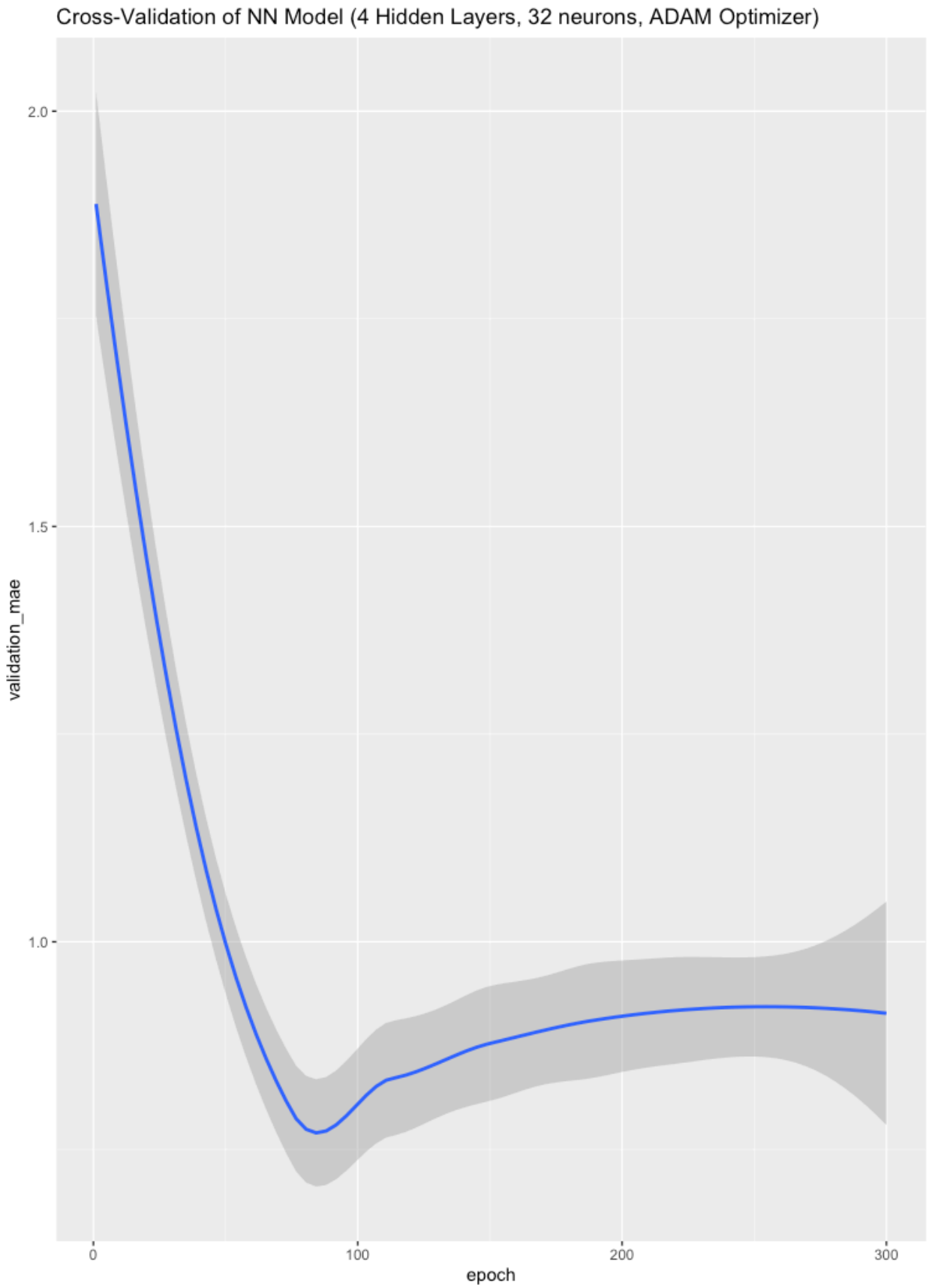


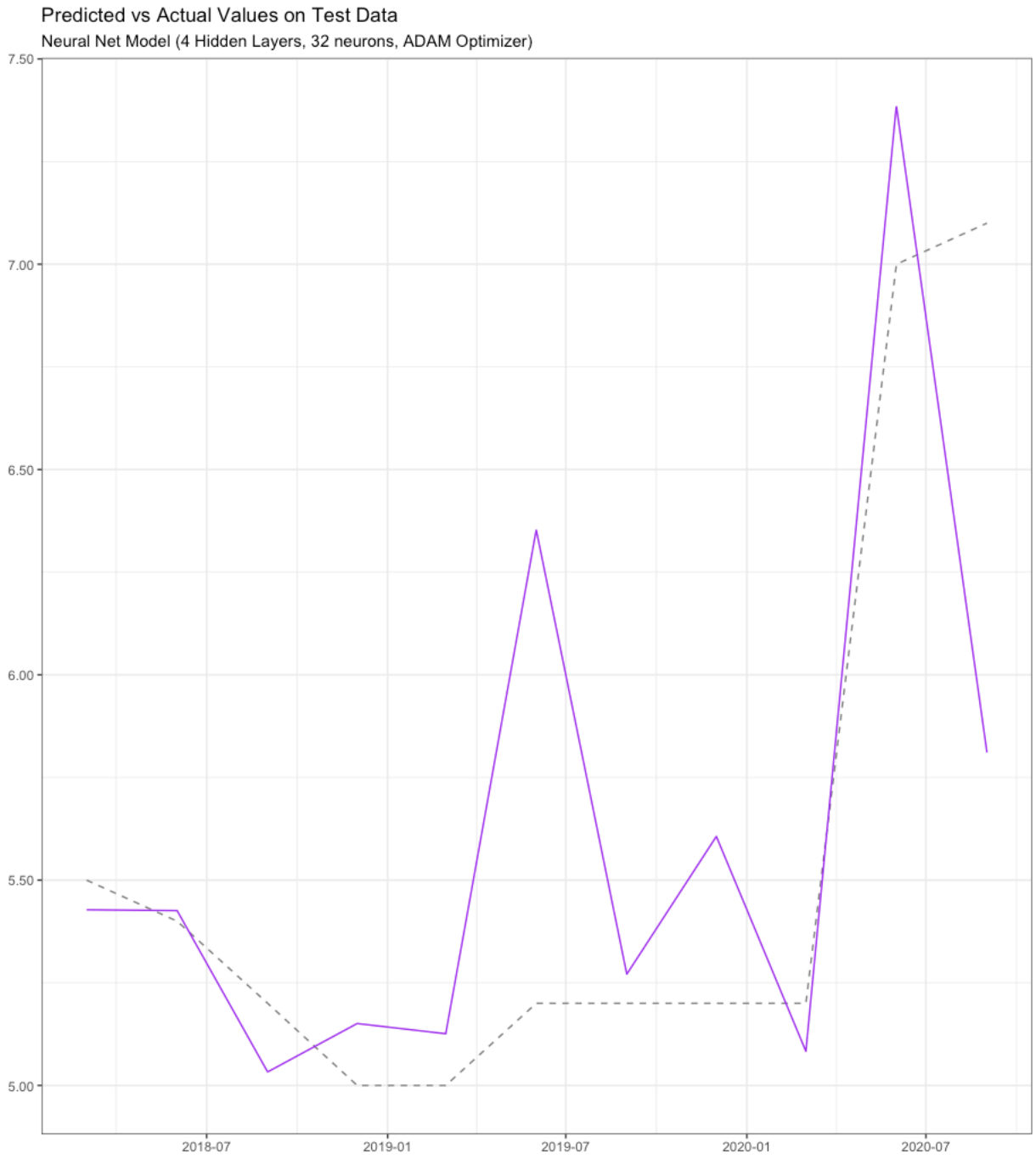
Figure 5: Multivariate analysis: Correlation plot

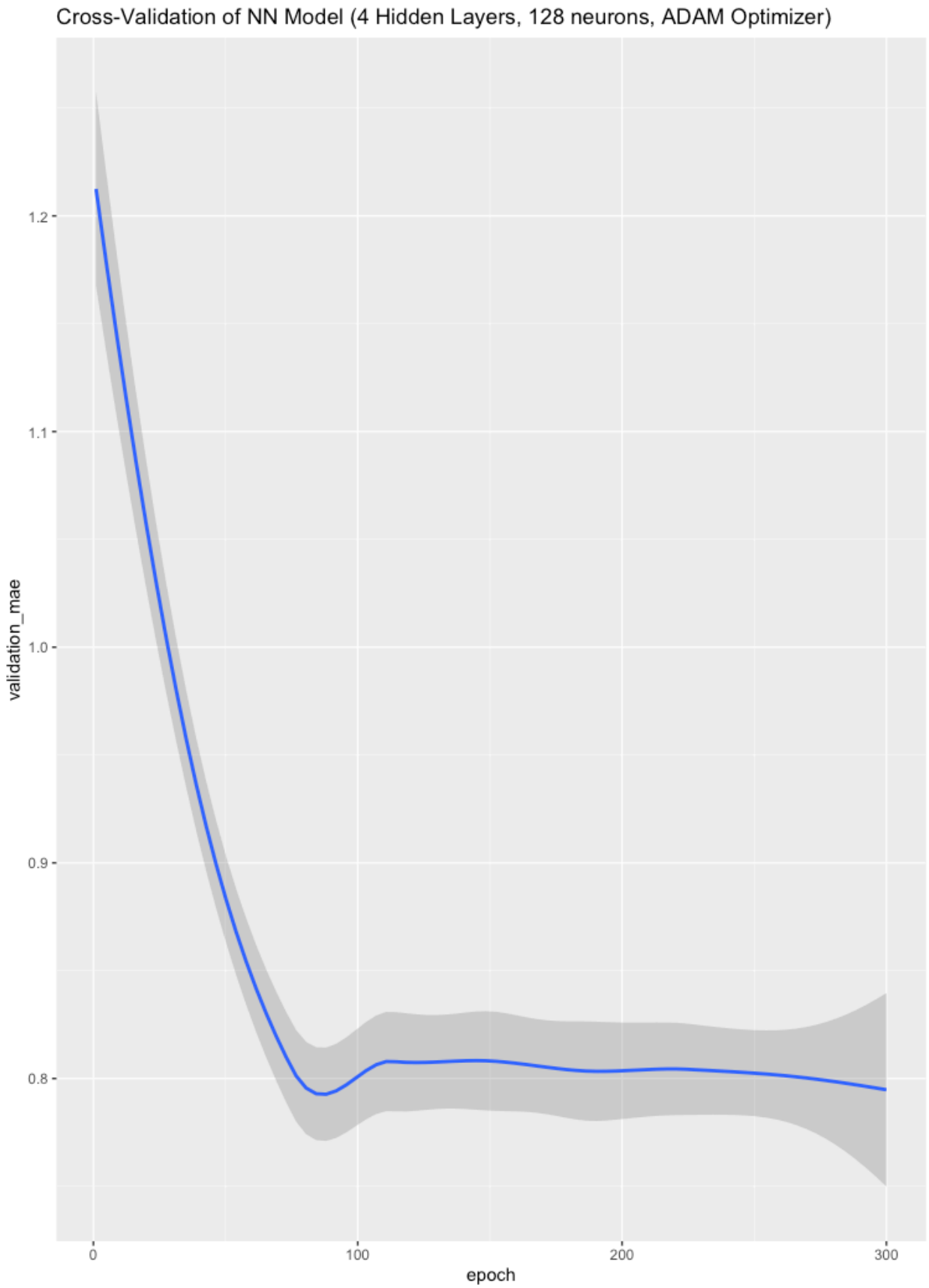

Predicted vs Actual Values on Test Data
Neural Net Model (3 Hidden Layers, 64 neurons, ADAM Optimizer)
7.00
6.50
6.00
5.50
5.00
Unimployment Rate, %
2018-07
2019-01
2019-07
2020-01
2020-07
Date

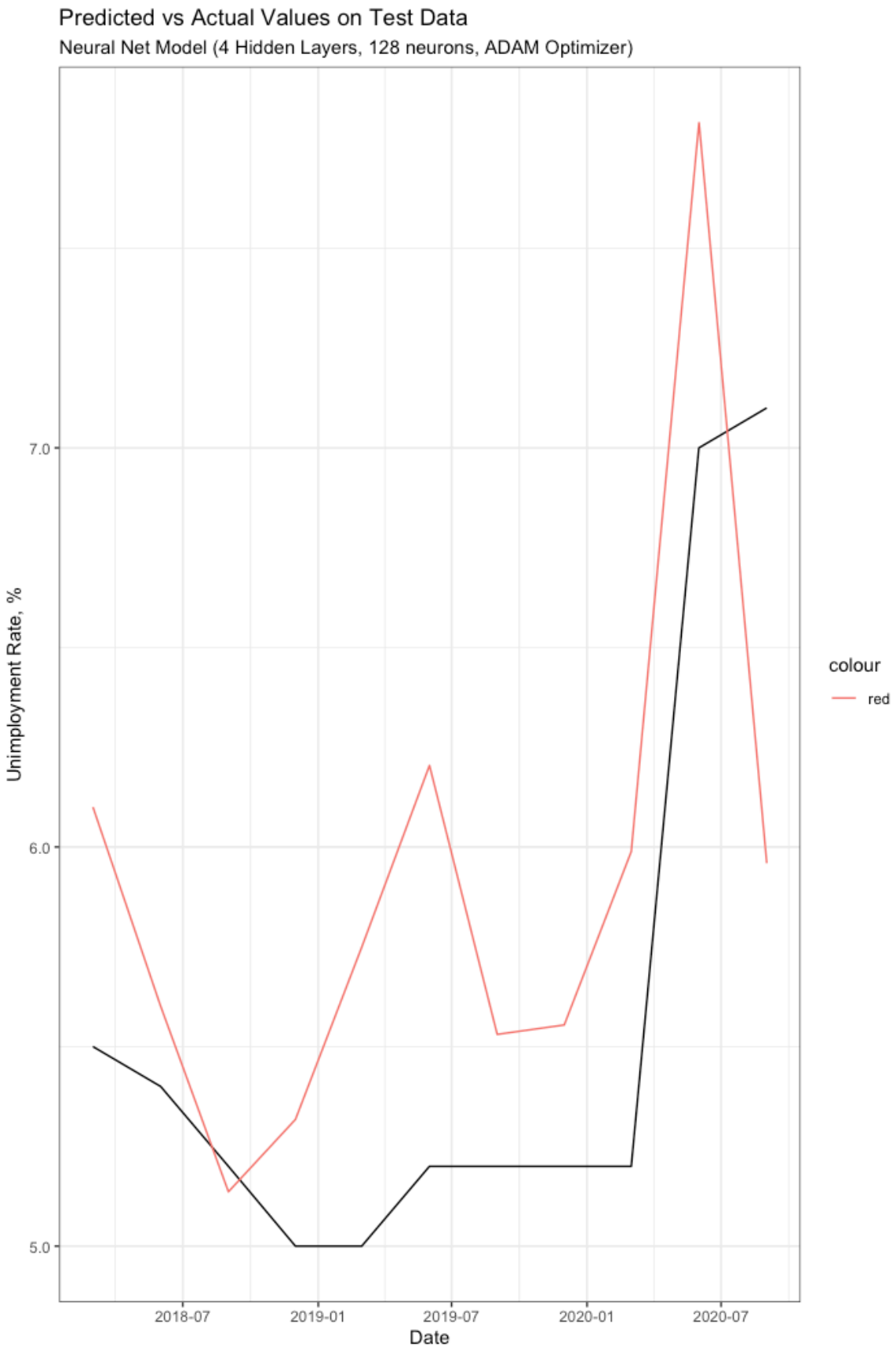

Cross-Validation of NN Model (3 Hidden Layers, 32 neurons, ADAM Optimizer)
validation_mae
2.0
1.5
1.0
0
100
200
300
epoch

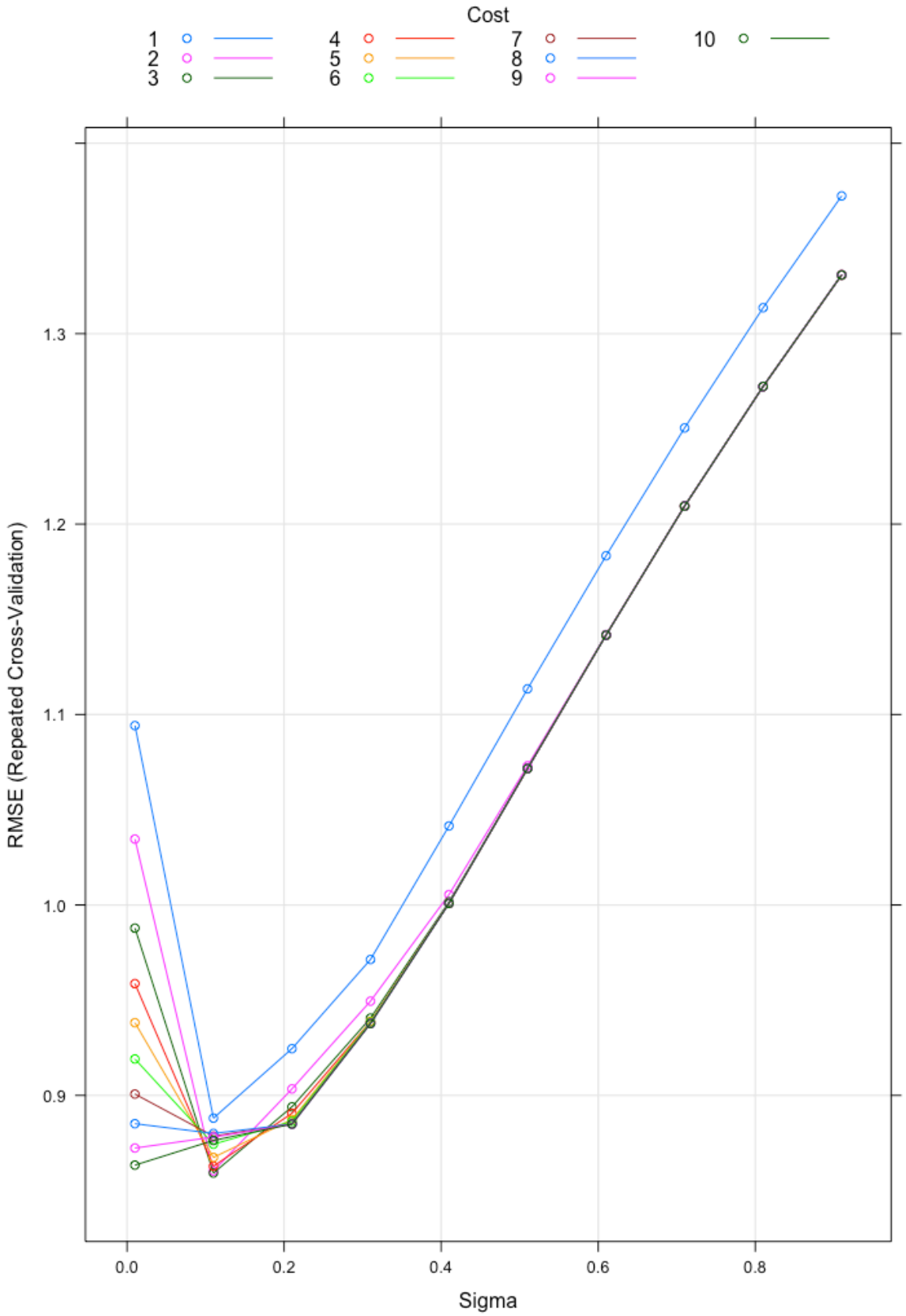

Predicted vs Actual Values on Test Data
Neural Net Model (3 Hidden Layers, 32 neurons, ADAM Optimizer)
7.0
6.0
5.0
2018-07
2019-01
2019-07
2020-01
2020-07
colour
purple

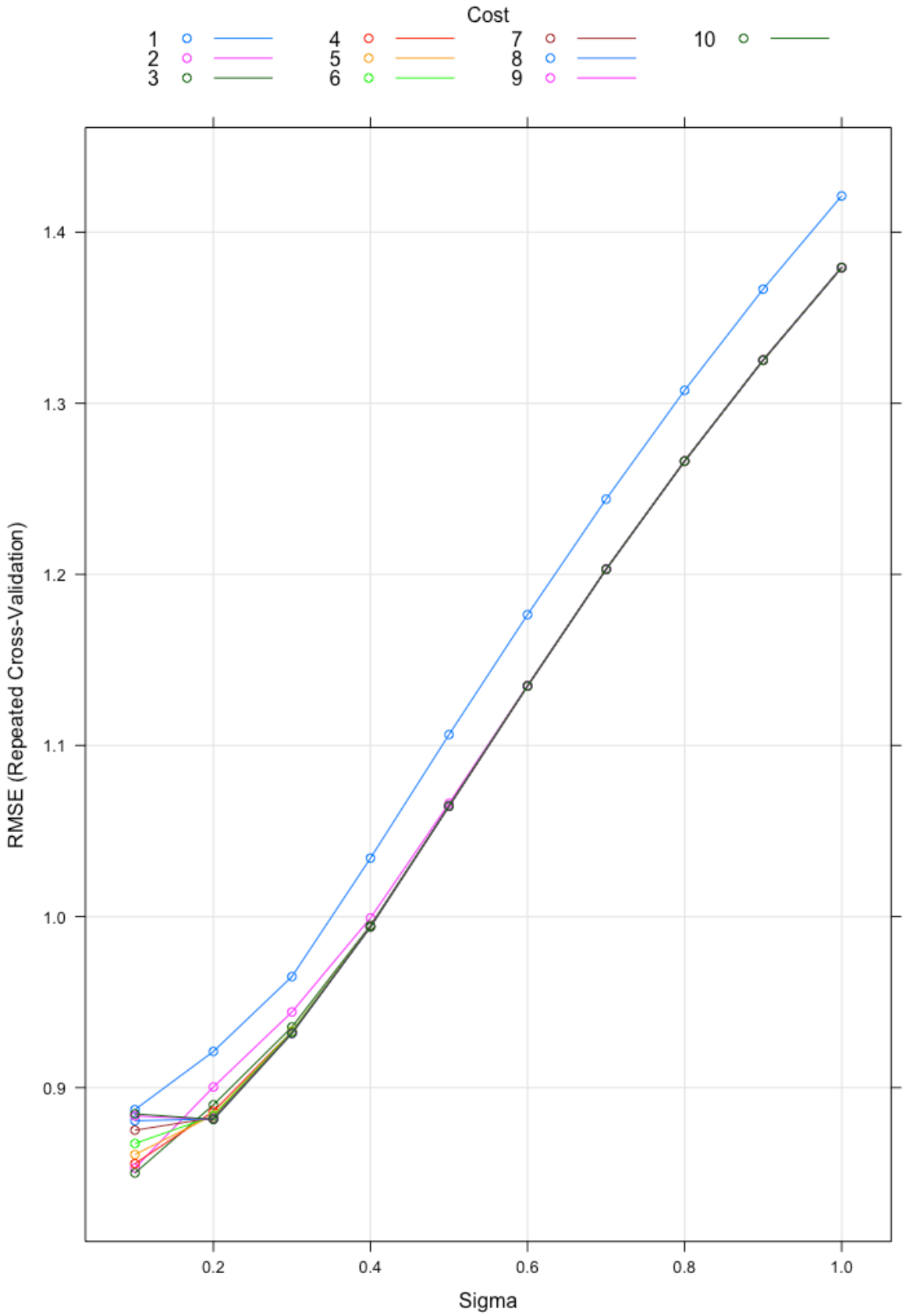

Cross-Validation of NN Model (4 Hidden Layers, 32 neurons, ADAM Optimizer)
validation_mae
2.0
1.5
1.0
0
100
200
300
epoch


Predicted vs Actual Values on Test Data
Neural Net Model (4 Hidden Layers, 32 neurons, ADAM Optimizer)
7.50
7.00
6.50
6.00
5.50
5.00
2018-07
2019-01
2019-07
2020-01
2020-07


Cross-Validation of NN Model (4 Hidden Layers, 128 neurons, ADAM Optimizer)
validation_mae
1.2
1.1
1.0
0.9
0.8
0
100
200
300
epoch


Predicted vs Actual Values on Test Data
Neural Net Model (4 Hidden Layers, 128 neurons, ADAM Optimizer)
Unimployment Rate, %
7.0
6.0
5.0
2018-07
2019-01
2019-07
2020-01
2020-07
Date
colour
red


Cost
1
2
3
4
5
6
7
8
9
10
RMSE (Repeated Cross-Validation)
1.3
1.2
1.1
1.0
0.9
0.0
0.2
0.4
0.6
0.8
Sigma


Cost
1
2
3
4
5
6
7
8
9
10
RMSE (Repeated Cross-Validation)
1.4
1.3
1.2
1.1
1.0
0.9
0.2
0.4
0.6
0.8
1.0
Sigma

References

Abu-Mostafa, Yaser S. 1986. "Neutral Networks for Computing?" In *AIP Conference Proceedings*, 151:1–6. 1. American Institute of Physics.

Australia, Year Book. 2008. "Australian Bureau of Statistics." *Canberra, Australia* 161.

Awad, Mariette, and Rahul Khanna. 2015. "Support Vector Regression." In *Efficient Learning Machines*, 67–80. Springer.

Bayar, Yilmaz, and others. 2016. "Financial Development and Unemployment in Emerging Market Economies." *Scientific Annals of Economics and Business* 63 (2): 237–45.

Biddle, Nicholas, Matthew Gray, Maria Jahromi, Dinith Marasinghe, and others. 2020. "Changes in Paid and Unpaid Activities During the COVID-19 Pandemic: Exploring Labour Supply and Labour Demand."

Burda, Michael C, and Daniel S Hamermesh. 2010. "Unemployment, Market Work and Household Production." *Economics Letters* 107 (2): 131–33.

Carroll, Nick. 2006. "Explaining Unemployment Duration in Australia." *Economic Record* 82 (258): 298–314.

Cassidy, Natasha, Iris Chan, Amelia Gao, Gabrielle Penrose, and others. 2020. "Long-Term Unemployment in Australia| Bulletin–December Quarter 2020." *Bulletin*, no. December.

De Fontenay, Catherine C, Bryn Lampe, Jessica Nugent, and Patrick Jomini. 2020. *Climbing the Jobs Ladder Slower: Young People in a Weak Labour Market*. Australian Government, Productivity Commission.

Downes, Peter M, Kevin Hanslow, and Peter Tulip. 2014. "The Effect of the Mining Boom on the Australian Economy." *Reserve Bank of Australia Research Discussion Paper*, no. 2014-08.

Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep Learning*. Vol. 1. 2. MIT press Cambridge.

Gulli, Antonio, and Sujit Pal. 2017. *Deep Learning with Keras*. Packt Publishing Ltd.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2017. "The Elements of Statistical Learning Data Mining, Inference, and Prediction (12th Printing)." Springer, New York.

Horning, Ned. 2013. "Introduction to Decision Trees and Random Forests." *American Museum of Natural History* 2: 1–27.

Lingjun, He, Richard A Levine, Juanjuan Fan, Joshua Beemer, and Jeanne Stronach. 2018. "Random Forest as a Predictive Analytics Alternative to Regression in Institutional Research." *Practical Assessment, Research, and Evaluation* 23 (1): 1.

McCulloch, Warren S, and Walter Pitts. 1943. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *The Bulletin of Mathematical Biophysics* 5 (4): 115–33.

McDonald, Ian M. 2020. "Macroeconomic Policy to Aid Recovery After Social Distancing for COVID-19." *Australian Economic Review* 53 (3): 415–28.

Moritz, Steffen, and Thomas Bartz-Beielstein. 2017. "imputeTS: Time Series Missing Value Imputation in r." *R J.* 9 (1): 207.

Trafalis, Theodore B, and Huseyin Ince. 2000. "Support Vector Machine for Regression and Applications to Financial Forecasting." In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, 6:348–53. IEEE.

Wauters, Mathieu, and Mario Vanhoucke. 2014. "Support Vector Machine Regression for Project Control Forecasting." *Automation in Construction* 47: 92–106.

Yang, Haiqin, Laiwan Chan, and Irwin King. 2002. "Support Vector Machine Regression for Volatile Stock Market Prediction." In *International Conference on Intelligent Data Engineering and Automated Learning*, 391–96. Springer.