MA5832-Assessment 3 (Capstone Project)

Student: Sacha Schwab

1 June 2020

Title:

Forecasting the Australian Unemployment Rate with Machine Learning and Deep Learning: A Learning Curve

Abstract

The COVID-19 crisis sets the development of unemployment rates in the spotlight of public interest. Pandemics are events that cannot be foreseen by any algorithm because they are so rare. However, it is important to understand the 'normal' evolution of unemployment in order to identify particular impacts of pandemics on employment, and what to do to improve the stability of the Australian labour market.

This report reflects my investigation into the evolution of the Australian unemployment rate using a machine learning algorithm and a neural network. The aim is to provide the basic model to effectively and efficiently predict the unemployment rate based on a set of predictors used by the Australian Bureau of Statistics.

The outcome is that neural networks do not perform as well as the gradient boosted decision trees selected for this assessment, in terms of RMSE of predictions. Neural networks require (by nature of complexity) significantly more computation time, and are not as interpretable as decision trees. However, as shown by the rather suboptimal predictive performance achieve, significant improvement will need to be achieved in further work dedicated to both methods. Such can be investigations into more sophisticated models in deep learning (e.g. LSTM), and attempts in other decision tree models. Also, improvement of the data may result in improved predictions.

The Australian employment market has a distinct structure, which can be seen in e.g. the importance of the mining sector, the particular predominance of male or female employment in certain industries, or the impact of work conditions (e.g. those leading to more flexibility). Hence, additional data may be (a) industry-related performance figures such as large project status or industry-specific imoport/export, (b) overall working hours, job turnover rates, or (c) changes in female / male employment.

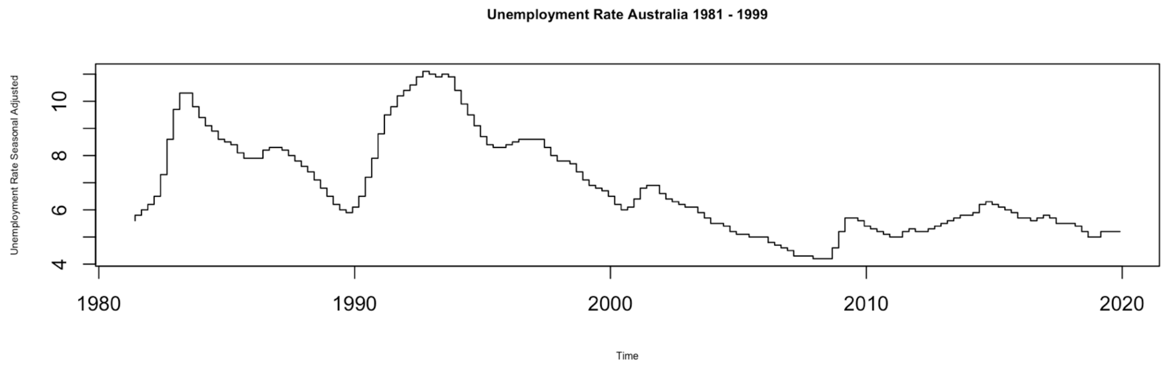
1. Introduction

1.1 Unemployment in general

Unemployment can have many sources, such as (see Wikipedia on "Unemployment"), e.g. structural changes such as new technologies and inventions, status of the economy, government policies, regulation and market.

There are multiple correlations between economic, social-cultural factors and unemployment rate seen as relevant by the literature. As an example, there is a clear relationship between general economic growth and unemployment (see e.g. Furhmann 2020).

1.2 Unemployment Rate in Australia 1999 - 2019



After major peaks in the 1980s and 1990s (as seen in the plot above) due to different reasons (see Fahrer and Heath), the unemployment rate from 1999 - 2019 fluctuated at around 6.9 and 4.2 percent.

My online and literature showed that, among many, some of the general factors leading to non-seasonal changes in unemployment in Australia during 1999 - 2019 are (a) structural events in industries that have a significance in males (e.g. mining industry) or female workers; (b) changes in flexibility of work arrangements; (c) changes in full-time education of young people (Gilfillan 2016), (d) general factors influencing the Australian economy such as commodity prices, exports, secondary effects like inflation deriving from decrease in employment and subsequent restraints in wage growth, real estate prices (Robinson 2015).

As these reasons are highly diverse, I will briefly reflect the development of the Australian unemployment rate by looking at phases reflecting various economic developments on a macro level.

Between 1999 and 2001 the Australian unemployment rate changed gradually from 7.1% to 6.0% and back to 7% between 1999 and 2001. One reason for the increase as from 2000 was a significant loss in full-time jobs.

After 2001 the unemployment rate saw a constant decrease to 4.2% in Q3 2008. The Federal Reserve Bank found that those changes in unemployment were much less significant than in the previous decades (Federal Reserve Bank 2010; see graph 2). This can be explained by a major upscale in flexibility in employment due to national policy changes (making it easier for firms and employees to negotiate work and pay arrangements) and increase in part-time jobs and a decline in average work hours (see Federal Reserve Bank 2010).

As from Q3 2008 Q3 2009 an increase in the unemployment happened, however, with +1.5% it was significantly lower compared to e.g. the US (+5.6%, see BLS 2012). This was due to the strong position of the Australian service sector, responses by the Federal Reserve Bank, and other factors (see Kennedy 2009).

In November 2013 the Reserve Bank of Australia considered that parallel to the increase in unemployment rate since mid 2011, the number of jobs actually increased significantly. It was observed that the labour force (the number of people working or available and looking for jobs) had not kept pace with the increase in employment, and that therefore the number of unemployed (and their share of the labour force) had increased. (Reserve Bank 2013).

From 2014 to 2019 the unemployment rate dropped at 1% at a more or less steady pace. The overall reasons for this may not have been reflected in the literature, however from various source at different points of time in this phase it is indicated that the Australian job market remained strong despite some downsizes in different industries such as properties and real estate.

1.3 Preliminary remarks on report structure

The structure of this report basically follows the requirements as provided in the assessment description and rubric. I have however added main parts of R codes, in particular those pertaining to machine learning and neural network models applied, but also a number of other code parts. This as per instruction by tutors provided in A3 assessment discussion board.

2. Data

2.1 Variables

The dataset for this assessment represents a sample of Australian national unemployment rates expressed in figures per quarters, from Q1 1981 to Q4 2019, as dependent variable, and a number of independent variables representing economic figures as described below. The data was collected from the Australian Bureau of Statistics (ABS). A total of 155 observations was collected.

The dependent variable i.e. the unemployment rate is presented as numerical variable.

The independent variables provide indications on a number of economic indicators such as (changes in) GDP (gross domestic product), government and all-industries final consumption rate, etc. They are all presented in numerical form.

According to indications provided by the tutors of this subject, the values are seasonal adjusted (as usually provided by statistics government agencies).

3 of the independent variables represent percentage changes, 4 variables contain absolute figures.

Below table shows the first 3 rows of the dataset as loaded from the file provided:

```
##        Month unemp_rate gdp gfce ifce  tti  cpi job_vac est_res_pop
## 1 1981-06-01        5.6 1.4  1.3  1.6  0.3 28.4    42.5     14923.3
## 2 1981-09-01        5.8 1.0 -0.8  1.0 -0.7 29.0    43.8     14988.7
## 3 1981-12-01        6.0 0.3 -1.0  0.7 -1.5 30.2    41.8     15054.1
```

Further indications on variables statistics such as distribution will follow after the thoghts on outliers and missing values detection and treatment.

2.2 Missing values

NA values, using R’s is.na() function, are identified for the variables job vacancies (5 NAs) and estimated Resident Population (2 NAs).

```
##       Month  unemp_rate         gdp        gfce        ifce         tti 
##           0           0           0           0           0           0 
##         cpi     job_vac est_res_pop        year       month     quarter 
##           0           5           2           0           0           0 
```

Missing data imputation:

In general, an extensive approach would be to impute the missing data based on customised machine learning models estimating the values (Duboue, p. 170). Such an approach is in my view only appropriate if we can assume that the affected variables have a close enough correlation with at least one of the other variables.

This may be the case for Job Vacancies, which might (from intuition) correlate with the Unemployment Rate. The Pearson correlation test indicates a p-value of 2.2e-16 for the relationship between the Unemployment Rate and Job Vacancies, which provides a positive message for using Job Vacancies for imputation of NAs in Job Vacancies. A simple linear regression proved to be rather successful with an RMSE of 0.115 of the mean of the test set. I therefore imputed the missing 5 values using this simple model.

As to the Estimated Resident Population however I would, from intuition, not see any other variable that would provide information for its prediction. Also in light of the fact that only 2 of the values are missing I will apply a simple approach, i.e. impute the mean of the previous 2 values in the time series.

The following limits the code display to filling the missing Job Vacancies values.

```r
fill.nas_linear <- function(data, col_NAs, col_estimate) {
  # Inputs the column with NA data, and the column(s) to use for regression estimate
  # Outputs the data with estimated values for NAs
  nas_index <- which(is.na(data[,col_NAs]))
  train <- data[-nas_index, ]
  model <- lm(train$job_vac ~ ., data = train)
  pred <- predict(model, data[nas_index, -col_NAs])
  data[nas_index, col_NAs] <- pred
  return(data)
}
data.clean <- fill.nas_linear(data.clean, 8, 2)
```

2.3 Distribution

The histograms as well as qqPlots provide that none of the independent variables has normal distribution, however some tendency to it.

For space real-estate reasons in this report I present these plots online under this link (https://github.com/fugu2/A3/issues/4#issue-640389292).

2.4 Outliers

X1 – X4 each have a small number (X1: 4, X2: 3, X3: 2, X4:5) of outliers. They might have some effect on the variance in scaling, however, from the histograms, they do not seem to be far off from the quartiles. For this reason and due to scarcity of the data and also in order to maintain the time series intact I have not removed them.

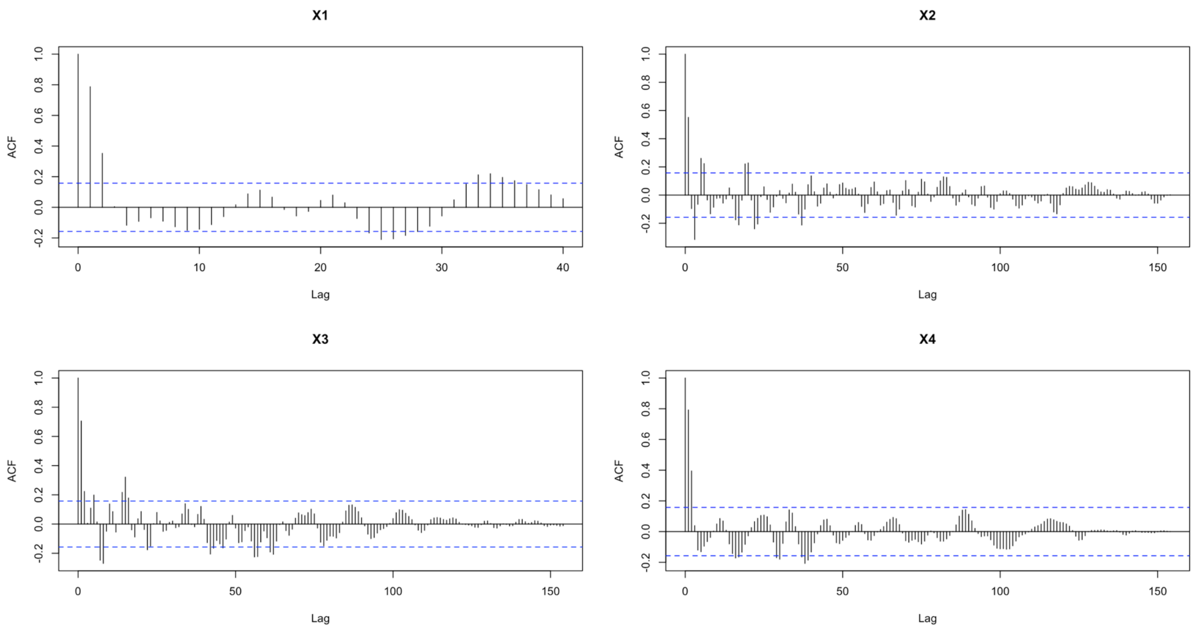
2.5 Patterns (seasonal, cyclic, trends)

From the visual there seems to be no significant trend in these data. Trend analysis plots using the acf() plot function from astsa library indicate that for e.g. X2 and X4 (and X1 over the period of a few years) some seasonality can be identified. However, the significant spikes are mostly (except for the first 3-4 lags) isolated in the range of the first 50 lags, in a range that is only slightly over significance. After this range all of the data do not bear any significant autocorrelation, which provides that they have already been adjusted in the data source (i.e. by the ABS).

X5-X7 have clear trends, which is not surprising, since

- the CPI has a “cumulative” shape i.e. does not decrease, reflecting the inflation gradually increasing;
- job vacancies, according to ABS indications are not seasonally nor trend adjusted for the overall level (adjustment is only done by sector series)
- for the estimated resident population the absolute figures may be of more interest seen their political impact.

The following plots represent the ACF for X1-X4. ACF plots for the remaining variables can be seen under this link (https://user-images.githubusercontent.com/10763939/84903237-23d23c00-b0ae-11ea-9fab-0023bf81dbb1.png).



## 2.6 Transformation

The variables are partly presented as differences (X1-X3), partly in absolute values. I have not found any literature as to whether such mixture is appropriate for models like Decision Trees and NN. However, I find that due to the character of the dataset as time series, the data feed should provide homogenously organized data in order to 'allow' the model to process the information from the data in an optimal manner.

Also, in a time series, information about the time step is relevant, i.e., for absolute values, the state at the beginning and at the end of the time step observation is presented, or, in one variable, the difference. Therefore, we need to either transform the difference variables each in two different absolute variables (e.g. starting with 100 as in 100%), or we replace the absolute values be the difference between the previous and the actual state. Alternatively there might be some intuition about cumulation, i.e. the absolute values as well as the differences would provide even more information. As I find interest the question whether this intuition can be confirmed, I produced three data cells (for verification by the models to be found as 'winners'): (c1) with only the differences (observation 1 from absolute values being 0), and (c2) with both differences and absolute values for all variables (starting with 100% for difference variables).

As recommended in a number of time series machine learning online sources, the time indications year and quarter are added and presented as categorical variables.

```r
# Building cells

add_differences <- function(data) {
  data$tti_diff <- 0
  data$cpi_diff <- 0
  data$vac_diff <- 0
  data$pop_diff <- 0

  for (i in 2:nrow(data)) {
    data$tti_diff[i] <- data$tti[i] - data$tti[i-1]
    data$cpi_diff[i] <- (100/data$cpi[i-1]*data$cpi[i]) - 100
    data$vac_diff[i] <- (100/data$job_vac[i-1]*data$job_vac[i]) - 100
    data$pop_diff[i] <- (100/data$est_res_pop[i-1]*data$est_res_pop[i]) - 100
  }
  return(data)
}
build_cell_1 <- function(data) {
  # cell 1 has the differences (i.e. convert columns 6-9)
  # first row is 100% (i.e. zero change)
  data <- add_differences(data)
  return(data[-c(6:9)])
}

build_cell_2 <- function(data) {
  # cell 2 has both differences and absolute values for all variables
  c1 <- build_cell_1(data)
  data$tti_diff <- c1$tti_diff
  data$cpi_diff <- c1$cpi_diff
  data$vac_diff <- c1$vac_diff
  data$pop_diff <- c1$pop_diff

  data$gdp_abs <- 100
  data$gfce_abs <- 100
  data$ifce_abs <- 100

  for (i in 2:nrow(data)) {
    data$gdp_abs[i] <- data$gdp[i] + data$gdp_abs[i-1]
    data$gfce_abs[i] <- data$gfce[i] + data$gfce_abs[i-1]
    data$ifce_abs[i] <- data$ifce[i] + data$ifce_abs[i-1]
  }
  return(data)
}

build_cell_3 <- function(data) {
  # cell 3 has c1 and y+1 as dependent variable
  data <- data.clean
  c3 <- build_cell_2(data)

  c3$unemp_rate_next <- 0

  for (i in 1:nrow(c3)-1) {
    c3$unemp_rate_next[i] <- c3$unemp_rate[i+1]
    # Based on internet search:
    c3$unemp_rate_next[nrow(c3)] <- 5.2
  }
  # Replace unemp_rate by next, and have unemp_rate as previous variable
  c3$unemp_rate_previous <- c3$unemp_rate
  c3$unemp_rate <- c3$unemp_rate_next
  c3 <- c3[ , -which(names(c3) %in% c("unemp_rate_next"))]
  return(c3)
}

c1 <- build_cell_1(data.clean)
c2 <- build_cell_2(data.clean)
c3 <- build_cell_3(data.clean)
```

2.7 Subsetting

As students are required to use the data from March 2015 (i.e. Q1 2015) – December 2019 for prediction i.e. as test set, I create a train and a test subset, accordingly (i.e. with the train set containing the data until Q4 2014).

```
split_data_excY <- function(data, split_month) {
  # Get numeric columns
  ind <- sapply(data, is.numeric)
  # Scale all except for output variable
  data[ind][,-1] <- lapply(data[ind][,-1], scale)
  train <- data[ which(data$Month < split_month), ]
  test <- data[ which(data$Month >= split_month), ]
  return(list(train, test))
}
```

3. Methodology

General Remarks

The data represent a time series, where Y at point of time t is denoted as

$$Y_t = S_t + T_t + \epsilon_t$$

And where S_t is a seasonal component, T_t is a trend and is an error term.

A large number of machine learning literature sources about forecasting time series provide that there is no prior knowledge about the independent values at point $t + 1$, whereas in this assessment the test set contains these. As a practical thought, the present assessment deals with the situation where e.g. the ABS has estimates about the future values of independent variables and intends to predict the unemployment rate based on these.

A more sophisticated approach would be it the output variable represents the future unemployment rate without knowing the actual independent values of the next quarter(s). Such would trigger the question of how long in the future (i.e. t until point i) the prediction shall be performed, as the input i.e. form of the data fed into the model building would be crucial (e.g. observation x contains the independent values at time X_t, whereas the corresponding Y would be represented by the unemployment rate or rates at $T_{t+1}, T_{t+2} \ldots T_{t+i}$).

As I find this approach interesting but dispose of limited time I selected a simple approach in order to gain a ‘flavour’ of unkown-future forecasting. Thus, I created another data cell (c3) containing the change-based variables from c1, add Y+1 as dependent variable, and treat Y as independent variable.

3.1 Machine Learning

3.1.1 Model choice

From the eligible machine learning models for this assessment it appears from Lin et al. (p. 165) that Support Vector (Machine) Regression may be a choice for non-large datasets. They propose (p. 175) even more however Random Forest (RF) models to achieve accuracy, stability and efficiency in time series prediction. Nielsen confirms that RF and Stochastic Gradient Boosted Models (GBM) can be used for time series. Winston, in his remarkable lecture video on boosting, provides that gradient boosting “doesn’t seem to overfit” even if the literature has not identified the reason for this (see video transcript).

For these reasons and as my first choice from intuition was to select GBM as an interesting option, I decided to use GBM for this part of the assessment.

3.1.2 Hyper-parameters

Hyperparameter selection for GBM is a challenge due to their extensive number the model implies. Ben Taieb et al. (p. 14) outline the importance of the shrinkage factor and the number of boosting iterations, which is actually a reference to Friedman 1999/1 (p. 1214) who also mentions that the number of terminal nodes as important parameter. Friedman provides that the shrinkage should be tuned “carefully”. In another paper (Friedman 2000, p. 5) he found that values <= 0.1 for the shrinkage parameter lead to a better generalization error.

Hastie et al. (p. 382) recommend a small number of nodes in the tree (i.e. that it is unlikely that >10 nodes will be required). They also recommend to use an early stopping procedure once performance on the validation dataset begins to degrade. As for subsampling, they refer to Friedman (1999) and confirm 1/2.

Ridgeway (p. 6) indicates that he uses a small as possible λ value. Interestingly he also mentions in the same paragraph that "Slower learning rates do not necessarily scale the number of optimal iterations".

For the loss function, Ridgeway (p. 7) mentions "distribution" (i.e. bernoulli for 0-1 outcomes) as the most important choice, with other choices for continuous outcomes to be "gaussian" and "laplace". As the latter is used for minimizing the absolute error, and I used "mae" in the neural network examination, I tested with mainly "laplace" distribution.

For the method for estimating the optimal number of iterations I selected "cv" out of the three choices as this is also the most similar approach to the one used for neural network (i.e. applying cross-validation).

As to the other parameters mentioned above I performed variations in an empirical manner as per below, as I find that this approach enhances my personal learning experience:

- Number of nodes per tree: Vary 2, 4 (if time left: 6); seen the limited number of variables my guess is that rather low numbers will do fine.
- Shrinkage parameter: 0.001, 0.005, 0.01, 0.05
- Number of trees: 100, 250, 500, 750
- Loss function: "laplace"

For the minimum number of observations in trees' terminal nodes I used the default of 10.

3.1.3 Implementation

I implemented the models using the "caret" library, with a 5-fold cross-validation with subsampling of, as mentioned above, 0.5.

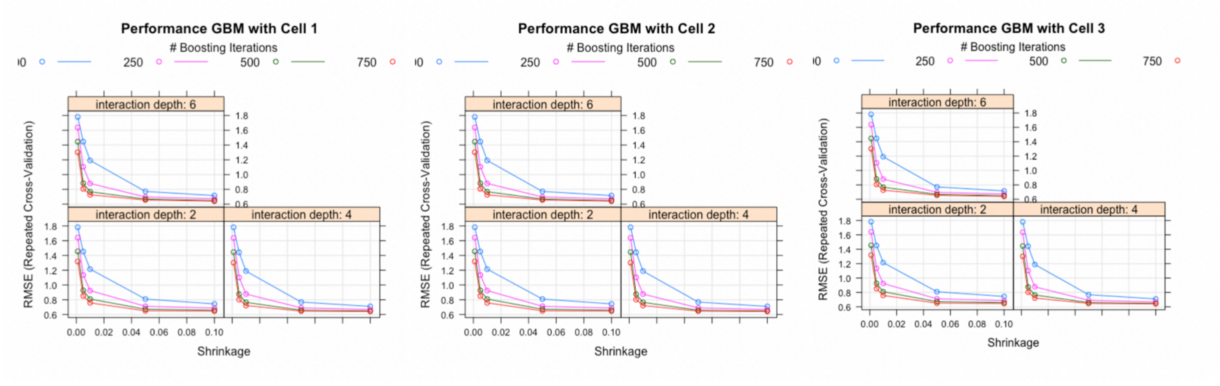
For each model variation for every data cell (see above) I computed the prediction using the test and subsequently registered the RMSE.

```
path <- "/Users/sachaschwab/Dropbox/JCU/09 Data Mining & Machine Learning/Assessments/A3/Results/"

run_gbm <- function(data, depth, shrinkage, n.trees, distribution, method) {
  tic("total time")
  # data <- c1
  # shrinkage <- 0.001
  # n.trees <- 500
  # depth <- 2
  # method <- "gbm"
  # distribution <- "laplace"

  train <- as.data.frame(split_data_excY(data, split.month)[1])
  test <- as.data.frame(split_data_excY(data, split.month)[2])
  train <- train[2:ncol(train)]
  test <- test[2:ncol(test)]

  set.seed(123) # set the seed to make your partition reproducible

  tc = trainControl(method = "repeatedcv", number = 5, repeats = 5, p = 0.5)

  boostgrid <- expand.grid(n.trees=c(n.trees), interaction.depth=c(depth),
                           shrinkage=c(shrinkage), n.minobsinnode = c(10))

  gbm <-  caret::train(unemp_rate ~ ., data=train,
                       method=method, distribution=distribution,
                       tuneGrid = boostgrid, trControl=tc, verbose = FALSE)

  pred <- predict(gbm, newdata = test)
  rmse <- rmse(test$unemp_rate, pred)

  # Write the results to file
  res <- data.frame("depth" = c(depth), "shrinkage" = c(shrinkage),
                    "n.trees" = c(n.trees), "distribution" = c(distribution),
                    "method" = c(method), "rmse" = c(rmse))

  write.table(res, paste0(path, "resultsGBM.csv"), col.names = FALSE,
              append = TRUE, sep = ",", row.names = FALSE)
  toc()
  return(rmse)
}

run_models <- function(data) {
  ns.trees <- c(100,250,500,750)
  shrinkages <- c(0.001, 0.005, 0.01, 0.05)
  depths <- c(2,4,6)
  for (n.trees in ns.trees) {
    for (shrinkage in shrinkages) {
      for (depth in depths) {
        run_gbm(data = data, depth = depth, shrinkage = shrinkage,
                n.trees = n.trees, distribution = "laplace", method = "gbm")
      }
    }
  }
}

# Plotting performance

get_models <- function(data) {
  set.seed(123) # set the seed to make your partition reproducible

  train <- as.data.frame(scale_and_split_data_excY(c1, split.month)[1])
  test <- as.data.frame(scale_and_split_data_excY(c1, split.month)[2])

  train <- train[2:ncol(train)]
  test <- test[2:ncol(test)]

  tc = trainControl(method = "repeatedcv", number = 5, repeats = 5, p = 0.5)

  boostgrid <- expand.grid(n.trees=c(100,250, 500, 750),
                           interaction.depth=c(2),
                           shrinkage=c(0.001, 0.005, 0.01, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4),
                           n.minobsinnode = c(10))
  gbmModel <-  caret::train(unemp_rate ~ .,
                            data=train,
                            method="gbm",
                            distribution="laplace", tuneGrid = boostgrid,
                            trControl=tc, verbose = TRUE)
  return(gbmModel)
}

gbModelc1 <- get_models(c1)
gbModelc2 <- get_models(c2)
gbModelc3 <- get_models(c3)
```

3.1.4 Performance and interpretation

The results for the different variations can be seen under this link (https://github.com/fugu2/A3/issues/6#issue-640513697).

Below plots provide an overview of the results:



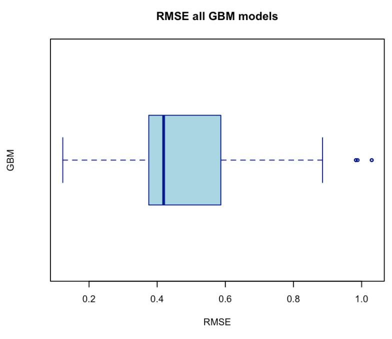
The results indicate that

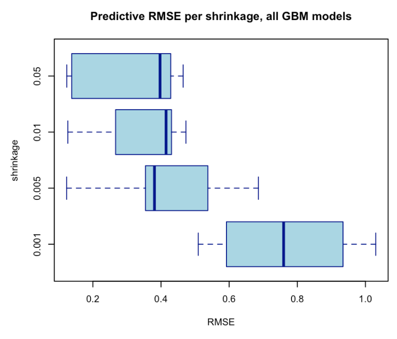
- there is no significant difference appearing with differing maximum nodes per tree. Value 4 for this parameter seems to produce very slightly better results. The full plot is not shown here

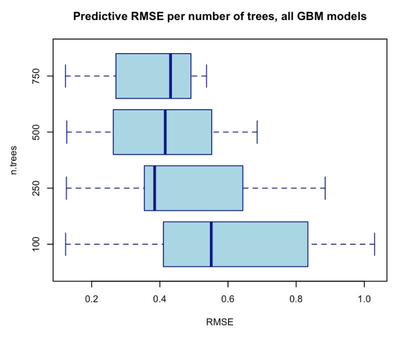
- increasing shrinkage value results in increase of CV RMSE. A closer look to the shrinkage, adding values 0.15, 0.2, 0.3 and 0.4 reveals that (a) Friedman's indication is indeed correct, i.e. that values > 0.1 do not provide significant better results, and (b) CV RMSE even increases when shrinkage value is beyond 0.1. See the plot for cell 1 as example: Link to the plot: https://user-images.githubusercontent.com/10763939/84733192-da92c700-af9d-11ea-9170-336b0af3b144.png (https://user-images.githubusercontent.com/10763939/84733192-da92c700-af9d-11ea-9170-336b0af3b144.png)
- growing number of trees also improves the training performance. Approximating 750 trees the significance however decreases. This might be explained by the fact that each tree is growing in sequence to fix the previous trees' mistakes (see Boehmke 2020, section 12), so with improving performance (minimizing the loss function), the variance of the sequence decreases.
- the number of trees has a more significant effect while the shrinkage is < 0.1. After that the performance lines of different numbers of trees tend to converge. This can be explained by Boehmke's observation (Boehmke 2020, section 12) that smaller shrinkage value increases "the risk of not reaching the optimum with a fixed number of trees", i.e., the smaller the value, the more accurate the model would be but requires more trees (in the sequence).
- training performance and variable importance do not change with the cells, which might be explained by the fact that the underlying values are the same, only with differing 'character'. For gradient boosting I did not assume that the different cells would provide significantly different results. If I could have expected differences, then probably for the training performance 'kicking in later' for c3 since it has the next quarter's unemployment rate as dependent variable. This was however not confirmed.

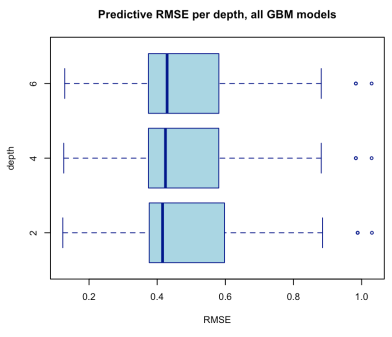
3.1.5 Predictive performance

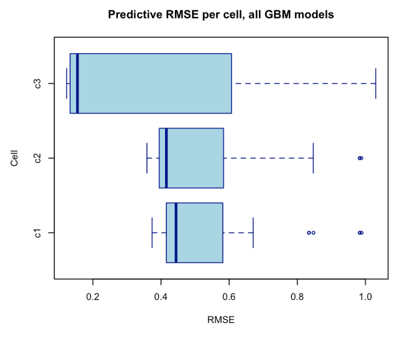
Below plots indicate the predictive RMSEs collected during training of the different model variations.









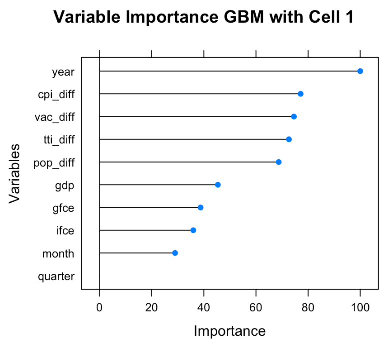


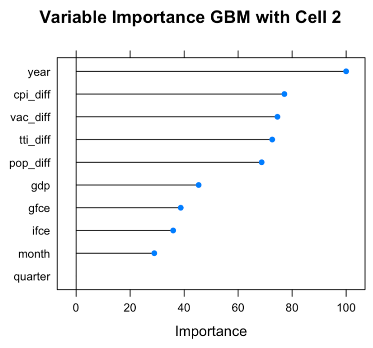
These and the detail list indicate that

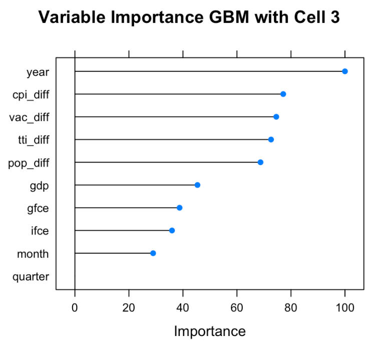
- lower and upper quartiles range between around 0.38 and 0.5, with lower extremes down to 0.12;
- increasing shrinkage value results in increase of CV RMSE. A closer look to the shrinkage, adding values 0.15, 0.2, 0.3 and 0.4 reveals that (a) Friedman's indication is indeed correct, i.e. that values > 0.1 do not provide significant better results, and (b) CV RMSE even increases when shrinkage value is beyond 0.1. See the plot for cell 1 as example, accessible under this link (https://user-images.githubusercontent.com/10763939/84733192-da92c700-af9d-11ea-9170-336b0af3b144.png)

- the number of trees is less important, however some performance gain can be noted between 250 and 500. This could be an indication for possible overfitting, or, to go with the finding by Winston (that stochastic gradient boosted trees do not overfit), reach of "maximum fitting" somewhere between the 250 and 500 trees.Also, this might be explained by the fact that each tree is growing in sequence to fix the previous trees' mistakes (see Boehmke 2020, section 12), so with improving performance (minimizing the loss function), the variance of the sequence decreases.
- there is no significant impoact coming from varying depth sizes. This confirms the rather low attention this parameter by the literature. My interpretation is that the objective of the gradient boosted tree algorithm to keep constructing trees with low bias favours low numbers of nodes;
- the number of trees has a more significant effect while the shrinkage is < 0.1. After that the performance lines of different numbers of trees tend to converge. This can be explained by Boehmke's observation (Boehmke 2020, section 12) that smaller shrinkage value increases "the risk of not reaching the optimum with a fixed number of trees", i.e., the smaller the value, the more accurate the model would be but requires more trees (in the sequence);
- against my intuition, data cell 3 performed significantly better than the other data cells. I have not found any reason or good interpretation for this "phenomenon". It is likely that based on the fact that the actual unemployment rate is included in the data, the differences between this and the next unemployment rate presented facilitate the prediction of the next rate (with unknown independent variables);
- most frequent model variations with RMSE < 0.3 (all for data cell 3 have shrinkage 0.05, 0.01, and 0.005;

Also, training performance and variable importance do not change with the cells, which might be explained by the fact that the underlying values are the same, only with differing 'character'. For gradient boosting I did not assume that the different cells would provide significantly different results. If I could have expected differences, then probably for the training performance 'kicking in later' for c3 since it has the next quarter's unemployment rate as dependent variable. This was however not confirmed.







3.2 Neural Network

3.2.1 Preliminary remarks on approach

As I am interested in empirical approaches, I applied such for the tuning NN.

Phase 1: I attempted to vary certain variables such as number of hidden layers, number of neurons per hidden layer, batch size. I collected the performance results as prediction RMSEs. For that part of the study, I erroneously used bootstrap instead of k-fold cross-validation, which in my view is however sufficient as for the next phase I used k-fold CV. Also, as seen below, this mistake ended in some interesting insight into the effects of different validation approaches.

Phase 2: From performance results analysis, I selected combinations that would possibly produce promising results and enhanced performance evaluation only for these, and validation of predictions deriving from epoch numbers where metrics would not significantly decrease any more (i.e. where overfitting would start – see Trinh et al., week 6 subject material).

3.2.2 Network structure

As noted above I basically implemented an experimental approach to estimate the best number of hidden layers and number of neurons, using indications from Trinh et al. (week 6 material), and other sources.

The basic structure of the neural network that I implemented is a vanilla NN with input layer of the size of the variables used for training, 1-4 hidden layers, and an output layer with 1 cell. I selected vanilla NN since from indications from tutors to this subject in week 5 online collaboration session, more sophisticated networks are not recommended.

3.2.3 Hyperparameters

- Activation function: I selected Relu for each of the input and hidden layers as activation function in order to keep the model computationally efficient (due to my limited GPU resources). From online reviews (see e.g. Missinglink) a “start” Relu as a reliable option should suffice.
- Optimizer: From literature review (see e.g. Khandelwal 2019) Adam optimizer appears to be a valid choice since it combines other optimizers (Adagrad, RMSprop), leveraging their advantages, and mitigating certain deficiencies (e.g. reducing diminishing learning rates from Adagrad). I therefore selected “adam” optimizer parameter.
- Loss: As the “standard” choice for regression problems appears to be MSE (see Brownlee on loss functions) and outliers do not seem to be an issue in the dataset (for which MAE would be better) I selected MSE as loss function.
- Monitoring metrics: Despite the loss function being MSE I selected MAE for my own learning experience.
- Since my approach was to empirically investigate in different set-ups for hidden layers, numbers of neurons and batch sizes I selected to not use x-fold cross-validation for this part in order to save compuation time, i.e. limited to bootstrap validation (using the “validation_split” parameter, see Trinh et al. 2020, week 6 material), with a split of 1/3.

Regaring possible overfitting issues, as the dataset is quite small, I did not apply any of the solutions such as regularization, dropout etc.

3.2.4 Scaling

As advised by tutors in week 6 discussion board to this subject, I applied MinMax-scaling to the independent variables for training.

3.2.5 Code

The following code lines represent the status at the time of the investigation into the training performance. As mentioned above, for the first (empirical phase) the difference was that I used (a) loops to produce outputs for all variations, and (b) bootstrap validation.

```r
minmax <- function(x) {
  return((x- min(x)) /(max(x)-min(x)))
}

run_nn_cv <- function(data, mode, num_epochs, batch_size, num_hidden_layers, num_neurons, k) {
  # Subset
  tic()
  train <- as.data.frame(split_data_excY(data, split.month)[1])
  test <- as.data.frame(split_data_excY(data, split.month)[2])
  train <- train[2:ncol(train)]
  test <- test[2:ncol(test)]
  train_data <- train[,2:ncol(train)]
  train_y <- train$unemp_rate
  test_data <- test[,2:ncol(test)]
  test_y <- test$unemp_rate

  train_data <- apply(train_data, 2, function(x) (x- min(x)) /(max(x)-min(x)))
  test_data <- apply(test_data, 2, function(x) (x- min(x)) /(max(x)-min(x)))

  # Matrices for tensors
  train_data <- as.matrix(train_data)
  train_y <- as.matrix(train_y)
  test_data <- as.matrix(test_data)
  test_y <- as.matrix(test_y)
  indices <- sample(1:nrow(train))
  folds <- cut(indices, breaks = k, labels = FALSE)

  all_mae_histories <- NULL
  for (i in 1:k) {
    print("starting k")
    # Prepare the validation data: data from partition # k
    val_indices <- which(folds == i, arr.ind = TRUE)
    val_data <- train_data[val_indices,]
    val_targets <- train_y[val_indices]
    head(val_data)

    # Prepare the training data: data from all other partitions
    partial_train_data <- train_data[-val_indices,]
    partial_train_targets <- train_y[-val_indices]

    model <- keras_model_sequential() %>%
      layer_dense(units = num_neurons, activation = "relu", input_shape = c(ncol(test_data))) %>%
      layer_dense(units = num_neurons, activation = "relu") %>%
      layer_dense(units = num_neurons, activation = "relu") %>%
      layer_dense(units = 1)

    model %>% compile(
      optimizer = "adam",
      loss = "mae",
      metrics = c("mean_absolute_error")
    )

    # Train the model (in silent mode, verbose=0)
    history <- model %>% fit(
      partial_train_data, partial_train_targets,
      validation_data = list(val_data, val_targets),
      epochs = num_epochs, batch_size = batch_size, verbose = FALSE
    )
    print(paste0("finished k", i))
  }

  mae_history <- history$metrics$val_mean_absolute_error
  all_mae_histories <- rbind(all_mae_histories, mae_history)
  toc()
  return(all_mae_histories)
}

mae_history_c2_2layers_32neurons_1batch <- run_nn_cv(data = c2, mode = "validate",
                       num_epochs = 500, batch_size = 1,
                       num_hidden_layers = 2, num_neurons = 32, k = 2)
```

3.2.5 Empirical process (phase 1)

I ran first tests with the following combinations:

- number of neurons per hidden layer: 8, 16, 24, 32, 64, 128 neurons
- number of hidden layers: 1, 2, 3 and 4
- batch sizes with variations: 3, 6, 10, 15 and 20

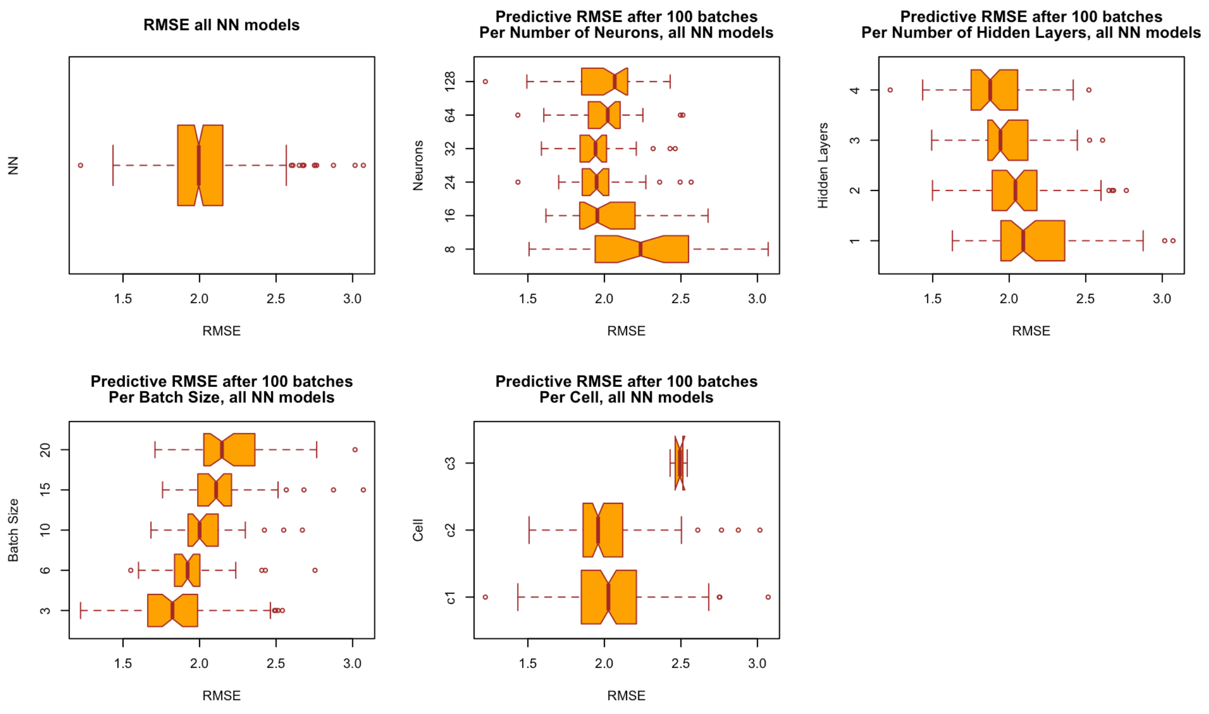
One of my basic thoughts was that due to the stochastic nature of the neural network training process (with random initial weights for the neurons) any run and further run would produce different performance results. Therefore, for stability / reproducibility of performance result I tried to apply seeds, which however did not succeed. I then came to the thought that it would be interesting to run the same model e.g. 10 times and take the mean of all resulting prediction RMSEs so to improve the overall indications compared to having the same seed for every run.

(I admit that this appraoch was only partly successful. However, it proved to be an asset since I was able to compare with k-fold cross-validation which produced much more robust results.)

Predictive performance of models produced in phase 1

From phase 1 (extensive tests using bootstrap validation), the following predictive indications were drawn:

```
## Warning in bxp(list(stats = structure(c(1.434562648, 1.849361186,
## 2.0263038895, : some notches went outside hinges ('box'): maybe set notch=FALSE
```



These plots indicate that

- numbers in the middle size (32 neurons) may lead to slightly better accuracy;
- the bigger the number of hidden layers, the better the performance; however, my “suspicion” is that the complexity of 4 hidden layers only leads to overfitting;
- smaller training batch sizes lead to better results;
- cells 1 and 2 provided similar results. As to cell 3 the results did not appear plausible.

Also, convergence in first tests varied at > 150 epochs, which let me define 500 epochs.

3.2.6 Performance of the produced NN models on the training dataset

In phase 2, from indications on the predictive performance in phase 1 I selected the following combinations for further investgations:

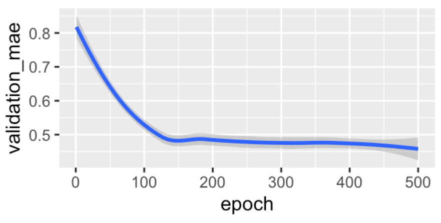
- Number of neurons per layer: 32, 64
- Hidden layers: 2 and 3

- Batch size: 3
- Data cells: c2 and c3

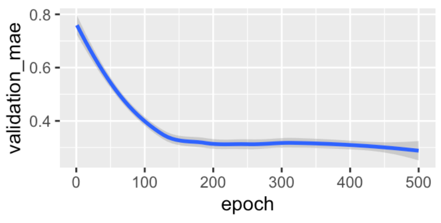
The plots below indicate that

- the training performance of the selected set-ups varies indeed in 4-fold cross-validation: The 2 models for cells 2 and 3 with 64 neurons per hidden layer perform significantly better than those with 32 neurons. This is interesting insofar as the number of variables is much less than both these numbers. It may be an indicator that there could be significant overfitting happening with increasing number of neurons.
- there is only little difference of the performance between the set-ups two different cells. The set-ups with 32 neurons per layer are almost identical, despite different approach in the dependent variable, which, for me, is surprising. I can only provide an educated guess about the reason, which might be that the combination of identical independent variables, and only slight differences in the dependent variables, lead to the weight finding a structure that lead to the almost same training performance curve.

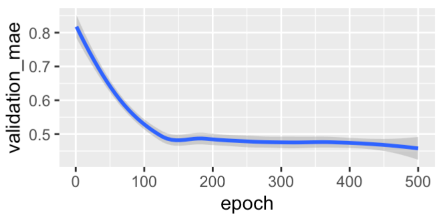
Cell 2, 32 Neurons/layer



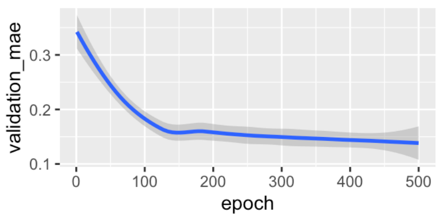
Cell 2, 64 Neurons/layer



Cell 3, 32 Neurons/layer



Cell 3, 64 neurons/hidden layer



3.2.7 Predictive performance on the test set (phase 2)

The following indicates the results after training with reduced number of epochs (145 after which overfitting would start).

(The full model code is available in the RMarkdown file due to its length and partly repetition of the code above.)

```
model_1 <- get_final_model(c2, num_neurons = 64, epochs = 145)
model_2 <- get_final_model(c3, num_neurons = 64, epochs = 145)
result_1 <- model_1 %>% evaluate(test_data, test_y)
result_2 <- model_2 %>% evaluate(test_data, test_y)
```

```
## [1] "Results final NN model 1 (c2, 2 hidden layers, 64 neurons per layer, 145 epochs:"
```

```
##                loss mean_absolute_error
##            1.477195            1.477195
```

```
## [1] "Results final NN model 2 (c3, 2 hidden layers, 64 neurons per layer, 145 epochs:"
```

```
##                loss mean_absolute_error
##            1.595807            1.595807
```

A deviation of 1.5% or 1.6% for the next quarterly unemployment rate is of course too much for a prediction of the unemployment rate. A statistical government agency would be embarassed if providing such. However, since neural networks are a very data-driven approach, further investigations should be undertaken in establishing additional independent variables. This following e.g. the particularities of the Australian labour structure, in terms of important industries, male/femal dominated industries, labour flexibility, etc. Also, more sophisticated networks should be tested, such as LSTMs.

5. Comparison

5.1 Cross-validated accuracy

The cross-validated accuracy for both models can get closer to each other as I would have expected, of course depending on the set-up, and with the reservation of possible overfitting (factor which would require more investigation).

This is, for the "winning" combinations, RMSEs of around 1.2 - 1.3 for GBM, and around 1.5/1.6 for NN. Still, as mentioned above in point 3.2.7, further enhancements should be achieved prior consideration of applying the models into practice.

5.2 Computational time

Computational time for training is significantly higher for NN than for GBM (X times). This of course due to the higher complexity (computational-intensive matrix calculations) of NN.

A 2-fold cross-validation of NN with 2 hidden layers, 32 neurons per layer, with c2 data, requires 155 seconds to compute on my machine, whereas for GBM the training process involving 750 trees, shrinkage of 0.001, requires 1.8 seconds.

5.3 Interpretability

Neural networks are per se only interpretable if separate "investigative" models are applied on top of the NN models. Entire studies are dedicated to the understanding of the mechanism of NNs (see e.g. Fan et al.). This due to the fact that the algorithms of how the weights are adjusted are known, however the actual "reason" of the "end-product", i.e. of how the weights trained model are composed to achieve the desired result, remains vastly unknown (see e.g. Molnar 2020, chapter 7), which makes the otherwise known as "blackboxes".

Decision Trees are "by nature" much more interpretable; however, for sophisticated algorithms such as gradient boosting, the result needs certain work to understand the contribution of each feature across the different trees, and it is hardly possible to understand the split logic of the model, reason for which they are often also called "blackboxes" (see e.g. Bonner 2018).

In my view lack of interpretability is however less important where there is less governance applied, i.e. where the humans applying the models are free to do so without having to explain the details mechanism of the model. This would not be the case for e.g. Anti-Money Laundering monitoring, where the regulator must understand the exact reason for the decisions made. See also indications in my assessment 2 report to this subject.

I imagine that for research reasons not only the performance of a model and accuracy of predicitons is important, but to understand the mechanism of the models in order to understand how the data work together to achieve the result, and in order to draw necessary conclusions for further data gathering. Of course also, this cannot be done without the help of domain experts.

6. Suggestions

As noted above, there are different ways to further improve the prediction of unemployment in Australia (if starting with the present dataset), such as considering

- particular influencing factors such as important industries that are sensitive to even slight structural changes or fluctuations in world market prices and import/export, such as mining;
- data on changes in employment of industries that have predominant male or female employment;
- the impact of work conditions (e.g. those leading to more flexibility).

Concret examples could be

a. industry-related performance figures such as large project status or industry-specific imoport/export,

b. overall working hours, job turnover rates,
c. changes female / male employment.

Also, of course, application of other models in both shown areas should be considered, such as, for NN, LSTMs (as these appear to be recommended for time series predictions) and different methods of decision trees, such as Extreme Gradient Boosting, Logistic Model Trees, Random Forest, with different parameters.

Conclusion and Findings

There is quite a way to go from here, and an amount of work to be undertaken until satisfactory results are achieved through machine learning and deep learning for prediction of Australian unemployment (starting with the given dataset). First, the diversified and clustered nature of the Australian economy and its particular mechnism require more thorough analysis of sectoral, regional and other clustering factors. Second, additional data will need to be provided. Third, the particular nature of time series requires addiitional thought into and sophistication of applied models and data structures. Fourth, for further research, keeping the model interpretable is important.

This has been a highly interesting and labour-intensive assignment, with the usual lack of happiness with the result in the end, but knowing that the process underwent not only many performance curves, but most importantly accelerate my learning curve tremendously, and has satisfied my interest in research insofar as the burden actually resulted in being able to draw interesting conclusions.

Appendix

References

- Ben Taieb, Sohaib et al. (2015). " Probabilistic time series forecasting with boosted additive models: an application to smart meter data" (research paper). Intitution: Monash University
- Biddie, Doug et al. (2001), “The Australian Labour Market 2001”. The Journal of Industrial Relations, pp 171-195. Editor: William Mitchell, Melbourne
- BLS, US Bureau of Labor Statistics (2012). “The Recession of 2007–2009” (government publication). URL: https://www.bls.gov/spotlight/2012/recession/pdf/recession_bls_spotlight.pdf (https://www.bls.gov/spotlight/2012/recession/pdf/recession_bls_spotlight.pdf)
- Boehmke, Bradley (2020). “Hands-On Machine Learning with R” (online publication, self-edited). URL: https://bradleyboehmke.github.io/HOML/ (https://bradleyboehmke.github.io/HOML/)
- Bonner, Will (2018). “Understanding gradient boosted model predictions” (internet article). URL: https://blog.marketfinance.com/2018/06/29/understanding-gradient-boosted-model-predictions/ (https://blog.marketfinance.com/2018/06/29/understanding-gradient-boosted-model-predictions/)
- Brownlee, Jason (2019). “How to Configure the Number of Layers and Nodes in a Neural Network” (blog). URL: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/ (https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/)
- Brownlee, Jason (2019). “Time Series Forecasting as Supervised Learning” (blog). URL: https://machinelearningmastery.com/time-series-forecasting-supervised-learning/ (https://machinelearningmastery.com/time-series-forecasting-supervised-learning/)
- Brownlee, Jason (2019). “How to Choose Loss Functions When Training Deep Learning Neural Networks” (blog). URL: https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/ (https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/)
- Cryer, Jonathan D. and Chan, Kung-Sik (2008). “Time Series Analysis with Applications in R” (book). Publisher: Springer.
- Duboue, Pablo (2020). “The Art of Feature Engineering” (book). Publisher: Cambridge University Press

- Federal Reserve Bank (2013). “Statement in Monetary Policy - November 2013, Box B: The Increase in the Unemployment Rate” (government publication). URL: https://www.rba.gov.au/publications/smp/2013/nov/box-b.html (https://www.rba.gov.au/publications/smp/2013/nov/box-b.html)
- Federal Reserve Bank (2010). “Bulletin - March 2010, The Labour Market during the 2008-2009 Downturn” (government publication). URL: https://www.rba.gov.au/publications/bulletin/2010/mar/1.html (https://www.rba.gov.au/publications/bulletin/2010/mar/1.html)
- Fahrer, Jerome and Heath, Alexandra (1992). “The Evolution of Employment and Unemployment in Australia” (research discussion paper). Economic Research Department, Reserver Bank of Australia
- Friedman, Jérôme (1999), “Greedy Function Approximation: A Gradient Boosting Machine“ (research paper). Stanford University
- Friedman, Jérôme (2000), “Stochastic Gradient Boosting“ (research paper). Stanford University
- Furhmann, Ryan (2020). “Okun’s Law: Economic Growth and Unemployment” (web article). URL: https://www.investopedia.com/articles/economics/12/okuns-law.asp (https://www.investopedia.com/articles/economics/12/okuns-law.asp)
- Gilfillan, Geoff (2016), “Developments in the youth labour market since the GFC” (research paper summary). URL: https://www.aph.gov.au/About_Parliament/Parliamentary_Departments/Parliamentary_Library/pubs/rp/rp1617/Yout (https://www.aph.gov.au/About_Parliament/Parliamentary_Departments/Parliamentary_Library/pubs/rp/rp1617/You
- Kennedy, Steven (2009). “Australia’s response to the global financial crisis” (government article). The Australian Treasury. URL: https://treasury.gov.au/speech/australias-response-to-the-global-financial-crisis (https://treasury.gov.au/speech/australias-response-to-the-global-financial-crisis)
- Khandelwal, Renu (2019). “Overview of different Optimizers for neural networks” (online blog). URL: https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3 (https://medium.com/datadriveninvestor/overview-of-different-optimizers-for-neural-networks-e0ed119440c3)
- Lin, Lin et al. (2017). “Random forests-based extreme learning machine ensemble for multi-regime time series prediction” (research paper). In: Expert Systems With Applications (pp 164 176); School of Mechatronics Engineering, Harbin Institute of Technology, Harbin, China
- Missinglink.ai (2020). “7 Types of Neural Network Activation Functions: How to Choose?” (online blog). URL: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/ (https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/)
- Madsen, Henrik (2008). “Time Series Analysis” (book). Editor: Taylor & Francis
- Nielsen, Aileen (2019). “Practical Time Series Analysis: Prediction with Statistics and Machine Learning” (ebook). Editor: O’Reilly Media
- OECD Organisation for Economic Co-operation and Development (2020). “Unemployment rate forecast” (website). URL: https://data.oecd.org/unemp/unemployment-rate-forecast.htm (https://data.oecd.org/unemp/unemployment-rate-forecast.htm)
- OECD (2012). “Policy Priorities for International Trade and Jobs” (book). Editor: Douglas Lippoldt
- Ridgeway, Greg (2007). “Generalized Boosted Models: A guide to the gbm package” (package guide). Self-edited. URL: http://www.saedsayad.com/docs/gbm2.pdf (http://www.saedsayad.com/docs/gbm2.pdf)

- Robinson, Tim (2015). “Budget explainer: the forces influencing Australia’s economy” (press article). In: The Conversation online. URL: https://theconversation.com/budget-explainer-the-forces-influencing-australias-economy-41161 (https://theconversation.com/budget-explainer-the-forces-influencing-australias-economy-41161)
- Trinh, Kelly et al. (2020). Materials to subject MA5832 “Data Mining and Machine Learning”. James Cook University, Australia
- Winston, Patrick H. (unknown year). “Lecture 17: Learning: Boosting” (lecture video). Massechusetts Institute of Technology MIT. URL: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/lecture-17-learning-boosting/ (https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/lecture-videos/lecture-17-learning-boosting/)