Various Model Applications to Japanese Population Data for Variable Selection and Accuracy Improvement

Emily Mendez
STA 4241

Introduction

As time passes by, the population continues to exceed our expectations in multiple regions. The extreme growth of these countries can all be attributed based on specific variables such as births and deaths. The concept of a growing population is also quite similar to one of a decreasing population as the main contributors remain the same. Though in a decreasing population such as Japan, the death rate surpasses the birth rate (Figure 1.1). By definition, this case japan is experiencing would be known as a natural decrease. By contrast a natural increase would be when births exceed the amount of deaths. These factors have always been known for having a strong correlation with population density, but other than this are there more variables that can contribute to population growth with a high correlation? The subject of interest is to explore a decreasing population and determine which variables are affecting its population density and how accurately we can predict it.

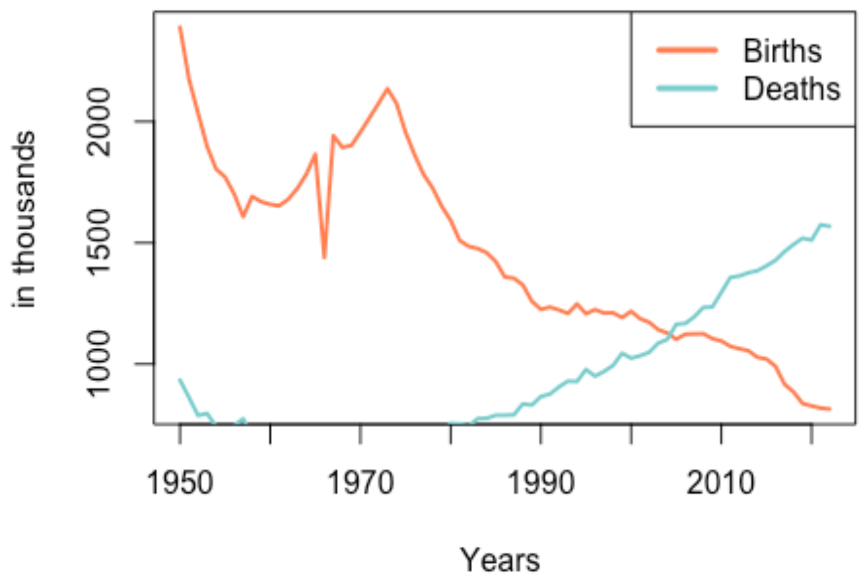

Figure 1.1: Japanese population data shows an increasing death rate overpassing the birth rate at approximately 2004.

This project specifically explored the Japanese population data from 1950 through 2022 to determine which factors affected the response variable, population density, by examining the relationships between the variables. Since most of the data is numeric the obvious choice for modeling the data and predicting variables would be a multiple linear regression. This method of regression is based on a supervised learning approach that matches each observation to a distinct predictor variable to measure the response. In addition to a linear model, we approach the problem of accurately predicting the quantitative response with various models. Other than linear regression, shrinkage and tree based models were applied to provide the linear regression with competitive prediction accuracies, and model interpretability. Throughout this project, we will be comparing each models choice for variable selection and observing how well each model performs in predicting population density through their respective MSE.

Data Collection and Preparation

This dataset was subsetted originally from the United Nations Department of Economics and Social Affairs world population prospect data [1]. The original data contained approximately 43 thousand observations with various countries listed and 55 indicators. By focusing on Japan with population density as our response, the dimensions were reduced to 73 observations and 55 variables. Some indicators included, but not limited to are, time, births by women aged 15 to 19, crude death rate, net number of migrants, total fertility rate, etc. As seen from some indicators listed, there are four main topics the variables are based upon. These topics are population, fertility, morality and migration. By using a linear regression with variables based on these four subjects to predict the quantitative response, multicollinearity was detected and variable selection began.

IndicatorNo	Topic	Indicator	Indicator Name	Unit
5	Population	PopDensity	Population Density, as of 1 July	persons per square km
14	Fertility	Births1519	Births by women aged 15 to 19	thousands
23	Mortality	CDR	Crude Death Rate	deaths per 1,000 population
53	Migration	NetMigrations	Net Number of Migrants	thousands

Table 1.1: Japanese population dataset indicators and their respective measurement units [1].

After loading the dataset, through R, I began preparations by removing the NA values. Fortunately, there was only one column that appeared to have NA's, column 13, with approximately 50% of the observations missing. Although, it was quite possible to perform imputations, I did not. This specific column represented the variable doubling time measured in years.

Investigating collinearity, is another useful and effective approach to preparing data. Assessing collinearity could help prevent the production of unreliable estimates and errors. With the use of a correlation matrix, we are able to find indicators of concern and have them removed or investigated for model building. Through observing the correlation matrix, I was able to discern high traces of multicollinearity. Variables among the highest correlations were the first couple indicators which were associated with total population from their description. It is then evident that this dataset will require a careful variable selection when creating a linear regression model, not so much for shrinkage models covered in the later part of the report as they perform their own variable selection. Due to the nature of a linear regression, it is quite possible to proceed with a forward, backward or stepwise selection paired with a subset selection. Alternatively, creating a function based off the correlation matrix could work as well.

The procedure that followed, data partitioning, allowed the dataset to be split into a training and testing set, 70 to 30 respectively. This allowed our model to be trained and tested in different sets to further improve how well it can predict our response and lower the training error. By having our dataset split, we were able to begin creating our models.

Models and Methodology

The methods applied in my project include a linear regression, an elastic net, a lasso and a decision tree model. The criteria used to select variables in my linear regression, was based off of lower correlations found in the correlation matrix. By creating a function to sum up how many predictors have moderate to high correlations above 0.65, I was able to eliminate variables with a high probability of making my linear model collinear. This process continued until I was left with enough variables to fit a model that rendered my variables significant. Eventually through continuously fitting and fixing, model 4 seemed to have significant pvalues, high adjusted r-squared and a low rse. The VIF and tolerance also seemed to be stable enough to diagnose the model to have little to no collinearity. In general, due to the dataset population density reaching its point of decline in 2010, the data visibly will turn out to have a decrease in slope after 2010 and a positive one from 1950 to 2010 (Figure 1.2). This in turn affects the normal qq plot and it's normality state. As shown in the plot, although the line shows little to no curvature, there is a moderate left skew (Figure 1.3). Therefore, the model would have been better fit if we investigated those specific years of ascent and descent separately to better interpret our variables and model.

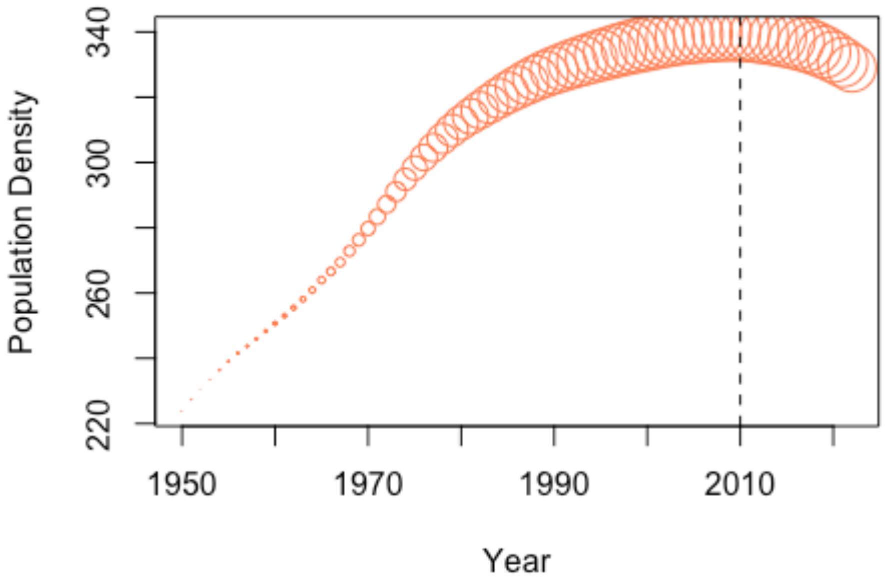

Figure 1.2: An increasing Japanese population density, by thousands, reaching its peak at 2010 then declining.

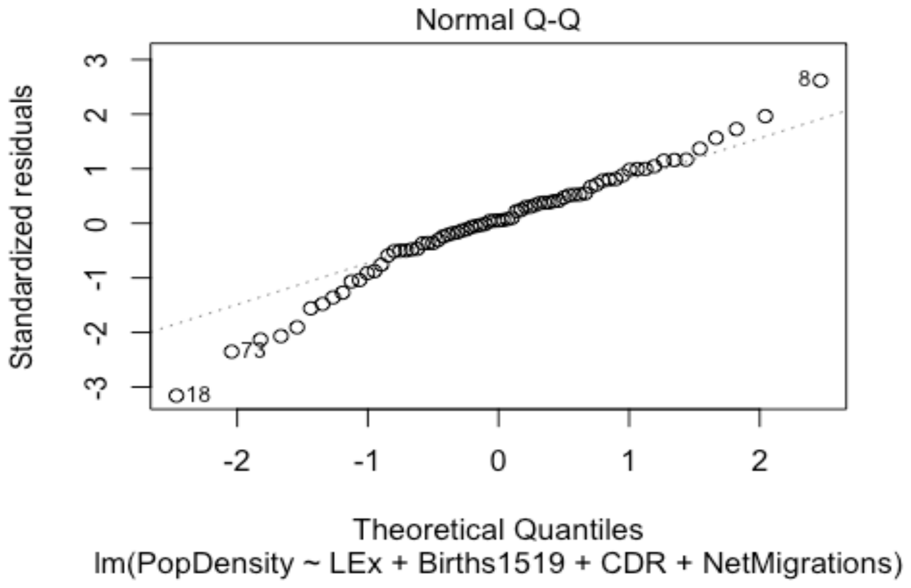

Figure 1.3: Normal QQ plot of the standardized residuals, with left skew.

The elastic net is a shrinkage model that utilizes variable selection, shrinkage parameters and uses grouping to select correlated variables to determine a good model. Although elastic net and lasso are both shrinkage models, many studies believe that the elastic net outperforms the lasso because of the way of handling correlated variables and combining both ridge and lasso methods. When fitting the model, we are assumed to use a method of tuning parameters; therefore, a 5 fold cross validation is enforced. These parameters, the lambda and alpha value, were found through resampling across the parameters. This specific method was used to provide a competitive advantage towards all other models in this project.

Lasso stands for least absolute shrinkage and selection operator as this method regularizes the $\hat{\beta}$ estimates towards zero or a value nearing zero and performs variable selection as well. Though at times when λ is large, we can expect the majority of the $\hat{\beta}$ coefficients to be zero. As stated before the lasso method also tends to choose variables randomly when correlated, so it under performs the elastic net method. As we utilize this method to attempt to reduce the RSS, we begin with standardizing our data, both training and test sets. Standardizing scales the data to have a mean of 0 and a standard deviation of 1. This in turn, allows us to measure the response variable as is with no shrinkage or changes occurring towards the β_0 coefficient. The data is then fitted into a glment function with an α value of 1, and returned with various lambda parameters in which the lambda.min is the choice for predicting.

Tree based models, are models used for simple interpretation at the cost of a higher MSE. In my project I utilized one tree that only represented a small portion of my dataset. In order to achieve a competitive MSE, it would be best to combine a large number of decision trees using either a random forest or a XGBoost. Both methods result in drastic improvements in prediction accuracy by learning from the previous trees. Although, I didn't use any large tree model, I used a simple decision tree to better interpret my findings in the variable selection.

Results

Results from my linear model show that the test error rate obtained from the MSE value was 16.26, my adjusted R^2 value was ≈ 0.99, the AIC $= 300.77$, and the BIC≈ 312.6. In all the variables that were deemed important to the linear model were, LEx, Births1519, CDR and NetMigrations. These variables and their respective descriptions are listed above in table 1.1. In terms of the training error which was 13.60, my model was overfit. Throughout these results most of the time the training error will be severely overfit as the training MSE is smaller than the testing MSE. The AIC in this case represents the fit of the model on the data, with the lowest number being the best fit. A result of 300 is definitely not great, but not that bad either, same goes for BIC.

The elastic net yielded a result of an adjusted R^2 of 0.99, a test error of 18.66 and a training error of 6.8. Therefore, this model was also overfit. Using a 5 fold cross validation, our final values of lambda and alpha were 0.073 and 1 respectively. In my elastic net model, it is shown that the MSE increases drastically as the $\log(\lambda) \rightarrow 2$ or 4 (Figure 1.4). Here my model determines the best α to be 1; therefore, the result is closest or equal to a lasso.

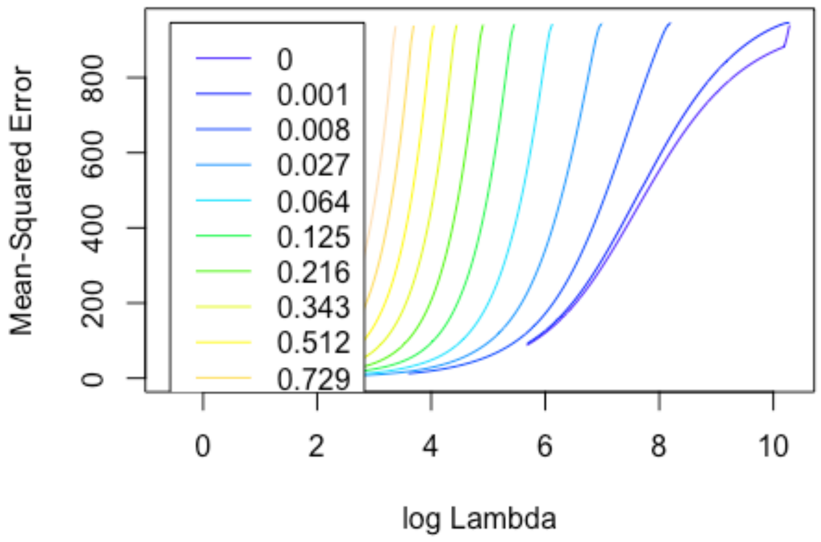

Figure 1.4: Alpha values of an elastic net model over log (λ); measured by MSE on the y axis.

The lasso method's variable selection, had selected 15 variables greater than zero. Variables that were deemed most significant are the features with the largest coefficients which include LE65, the life expectancy of at age 65, PopSexRatio, and Births1519. Another variable was CDR which was also included in my linear regression model. This model is overfitted as well, as the testing error is 12.12 and the training error is an extremely low value of 0.0014. The adjusted R^2 of this model was approximately around 0.9 which explains the approximately 90% of the variability in the mean population density.

The tree based model, decision tree, produced 5 nodes and selected 3 variables. These variables include LE65, PopSexRatio, and NRR; which also occur in the lasso variable selection. PopSexRatio represents the population sex ratio as of July 1st, and the NRR variable represents the net reproduction rate measured by surviving daughters per woman. The test MSE for this specific decision tree was 82.892 and a 37.16 for the training error, which is a lot higher than the other models we've seen. Despite the R^2 coming out to be higher than the lasso's at 0.94, this model along with all other models, was overfit.

The decision tree as shown in figure 1.5, begins with splitting the life expectancy at the age of 65 to 15 years. In this branch there are approximately 18 observations and the overall prediction was 257.3 in thousands for population density. This branch splits into node 4 and node 6/7 to determine the population density.

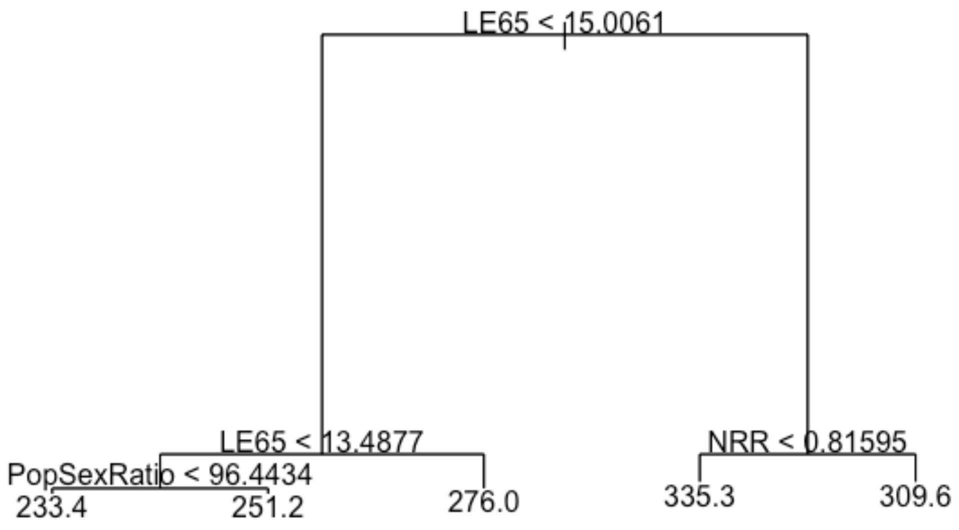

Figure 1.5: Decision tree of 5 terminal nodes and 3 variables.

Conclusion

Overall the model with the best test error rate is the lasso with a MSE of 12.12 (Table 1.2). Alternately, the model with the highest test MSE was the decision tree model. The elastic net and linear regression models were quite of similar performance in all, and performed well. Variables that seemed to perform best in predicting the population density, were determined through the various variable selections that were often occurring. These variables were, LE65, PopSexRatio, and Births 1519. Observing these predictors indicates that there is little to no collinearity between these three as they all are taken from different topics. With LE65, PopSexRatio, and Births 1519 coming from morality, population and fertility subjects respectively.

Models	**Linear Regression**	**Elastic Net**	**Lasso**	**Decision Tree**
R-Squared	0.989	0.9871	0.9018	0.9425
Test MSE	16.2586	18.6621	12.1219	82.891
Train MSE	13.601	6.801	0.001	37.160

Table 1.2: Final results of the fitting linear regression, Elastic Net, Lasso, and Decision tree models.

References

[1] United Nations, Department of Economic and Social Affairs, Population Division (2022).

World Population Prospects 2022, Online Edition.

Appendix

Loading in data

```
JapanPop=read.csv('/Users/admin/Documents/Csvv/japan.csv',header=TRUE)
JapanPop=JapanPop[,-13]
indicatorI=read.csv('/Users/admin/Documents/Csvv/Indicator.csv',header=TRUE)
knitr::kable(indicatorI[1:10,1:5],format="markdown")
```

Libraries

```
#loading in libraries
library(tree)

library(Metrics)

library(glmnet)

library(glmnetUtils)

library(MLmetrics)
```

Final Model from Linear Regression Analysis

```
#Fitting linear model from project part 2
lin.model=lm(PopDensity~LEx+Births1519+CDR+NetMigrations,data=JapanPop)
summary(lin.model)
```

Data Partitioning

```
set.seed(123)
trainInx=createDataPartition(JapanPop$Time,times=1,p=0.70,list=FALSE)
training=JapanPop[trainInx,]
testing=JapanPop[-trainInx,]
```

Fitting Linear model

```
#testing error
train.lin.m1=lm(PopDensity~LEx+Births1519+CDR+NetMigrations,data=training)
pred.lin.m1=predict(train.lin.m1,newdata = testing[,-6])
m1.MSE = mean( (pred.lin.m1 - testing$PopDensity)^2 )
m1.MSE
```

```
#training error
train.pred=predict(train.lin.m1,newdata=training)
train.MSE.1 = mean( (train.pred - training$PopDensity)^2 )
train.MSE.1
```

```
#summary of data info
sum.1=summary(train.lin.m1)
sum.1$adj.r.squared

## [1] 0.9893969

AIC(train.lin.m1)

## [1] 300.7741

BIC(train.lin.m1)

## [1] 312.5959
```

Elastic Net model

```
#removing top associated columns since I did the same in the previous project p
arts
train.elasNt=cva.glmnet(PopDensity~.,data=training[,-c(1:5)])

plot(train.elasNt)
```

```
set.seed(123)
#fitting elastic net model
elasticNet.fit=train(PopDensity~.,data=training[,-c(1:5)],method="glmnet",
                     trControl=trainControl(method = "cv",number=5))

#seeing which paramters the model chose
elasticNet.fit
```

```
#predicting the model with test data/ test error
elnet.pred=predict(elasticNet.fit,newdata = testing[,-c(1:5)])
m2.MSE = mean( (elnet.pred- testing$PopDensity)^2 )
m2.MSE
```

```
#training error
elnet.pred.tr=predict(elasticNet.fit,newdata = training[,-c(1:5)])
train.MSE.2 = mean( (elnet.pred.tr- training$PopDensity)^2 )
train.MSE.2
```

Lasso

```
#standardizing columns
set.seed(124)
trainingstd= apply(training[,-c(1:5)], 2, function(x){x/sd(x)})
testingstd = apply(testing[,-c(1:5)], 2, function(x){x/sd(x)})
```

```
#fitting a lasso model
lasso.fit = cv.glmnet(x=trainingstd[,-c(1)], y=trainingstd[,1], alpha=1,type.me
asure="mse")
```

```
#looking at the variable selection
print(coef(lasso.fit, s="lambda.1se"))
```

```
#predicting model with testign data
lasso.pred=predict(lasso.fit,s="lambda.min",newx = testingstd[,-c(1)])


#test error
m3.MSE= mean((lasso.pred - testingstd[,1])^2)
m3.MSE
```

```
#training error
lasso.pred.tr=predict(lasso.fit,s="lambda.min",newx = trainingstd[,-c(1)])
```

```
train.MSE.3= mean((lasso.pred.tr- trainingstd[,1])^2)
train.MSE.3
```

Decision Tree model

```
#fitting decision tree
tree.fit=tree(PopDensity~.,data=training[,-c(1:5)])
summary(tree.fit)
```

```
tree.fit

plot(tree.fit)
text(tree.fit,pretty=1)



pred.m1=predict(tree.fit,newdata = testing[,-c(1:5)])
m4.MSE=mean((pred.m1- testing$PopDensity)^2)
m4.MSE
```

```
#training error
pred.train=predict(tree.fit,newdata = training[,-c(1:5)])
train.MSE.4=mean((pred.train- training$PopDensity)^2)
train.MSE.4
```

```
#pruning info
cvTree.m1=cv.tree(tree.fit)
par(mfrow = c(1, 2))
plot(cvTree.m1$size, cvTree.m1$dev, type = "b",xlab="Number of Nodes",ylab = "C
ross Validation Error")
plot(cvTree.m1$k, cvTree.m1$dev, type = "b",xlab="K ",ylab = "Cross Validation 
Error")
```