National College of Ireland

Configuration Manual

MSc Research Project
Fintech

Chisom Nneamaka Ezeilo
Student ID: X18199798

School of Computing
National College of Ireland

Supervisor: Victor Del Rosal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Chisom Nneamaka Ezeilo

Student ID: X18199798

Programme: MSC. In Fintech **Year:** 2019-20

Module: Research Project

Lecturer: Victor Del Rosal

Submission Due Date: 17th August 2020

Project Title: EVALUATION OF THE IMPACT OF FINTECH PAYMENT SOLUTIONS OF THE GROSS DOMESTIC PRODUCT (GDP) OF DEVELOPING COUNTRIES IN SUB-SAHARAN AFRICA

Word Count: 1489 **Page Count:** 10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:

Date: 14/08/2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Chisom Nneamaka Ezeilo
Student ID: x18199798

1 Introduction

This research was carried out to identify and analyze the impact of fintech payment solutions on the GDP of developing countries. This configuration manual gives a detailed explanation of the steps that was required to carry out the analysis and to build the models used for the analysis. These steps have been organized in such a way to ease recreation of the project. Four regression models were built, and they include Partial Least Square Regression, Principal Component Regression, Support vector regression and Least Absolute Shrinkage and Selection Operator regression.

2 System Requirement

2.1 Hardware

- ✓ Windows OS version 10 Pro – 64bit
- ✓ Lenovo X230
- ✓ Processor: Intel(R) Core (TM) i5-3320M CPU @ 2.60GHz, 2.60 Mhz
- ✓ RAM: 8GB
- ✓ SSD: 237GB

2.2 Software

- ✓ Microsoft Excel 2016 – used for data wrangling and preprocessing
- ✓ R programming Language and R studio – Version 3.6.1 – used for data analysis
- ✓ Microsoft Word 2016 – this was used for report compilation

3 Data Selection & Integration

The Global Findex and Maddison data sets were used for the analysis. These datasets contained information from different regions, so the data selection process required the selection of countries in Sub-Saharan Africa (SSA) with the complete data for the two datasets.

- ✓ **Step 1**: Extract dataset from Global Findex[1] and Maddison Project[2].
- ✓ **Step 2**: Import the data into R Studio.
- ✓ **Step 3**: View the structure of the data and extract countries categories as SSA.

```
# importing the GDP dataset
```

[1] https://globalfindex.worldbank.org/
[2] https://www.rug.nl/ggdc/historicaldevelopment/maddison/releases/maddison-project-database-2018

```
maddison_GDP_dataset <- read.csv (file = 'maddison-data-gdp-per-capita.csv')

#  importing the global Findex Data
global_findex_dataset <- read.csv (file = 'Global Findex Database.csv')

# extracting rows with SSA countries
SSA_findex_data <- global_findex_dataset [copy_global_findex_dataset$Region == 'Sub-Saharan Africa',]

SSA_GDP_DATA <- maddison_GDP_dataset [maddison_GDP_dataset$Region == 'Sub-Saharan Africa',]
```

✓ **Step 4**: Evaluate the variables in the data to determine how to merge appropriately.

```
# checking the column names of the datasets
colnames(SSA_GDP_data)
colnames(SSA_findex_data)

# checking the number of variables in each
dim(SSA_findex_data)

[1] 104 813

dim(SSA_GDP_DATA)

[1] 2865    5
```

The Findex dataset has 813 columns hence it is essential to extract only columns related to Fintech payment.

✓ **Step 5**: Extract relevant column in dataset.

```
# 6. Extracting relevant fintech payment columns from Findex data

SSA_findex_data <- SSA_findex_data[,-c(6:170,183:606,609:682,696:733)]
```

✓ **Step 6**: Changing the data types of the variables.

```
#  Changing the type of the variables
SSA_findex_data$Year <- as. integer(SSA_findex_data$Year)
SSA_findex_data$Code <- as.character(SSA_findex_data$Code)
SSA_SSA_findex_data$Income.Level <- as. factor(SSA_findex_data$Income.Level)
SSA_findex_data$Region <- as.character(SSA_findex_data$Region)
SSA_findex_data$Income.Level <- as. factor(SSA_findex_data$Income.Level)

#Changing the variable type of GDP data set
copy_SSA_GDP_data$Code <- as.character(copy_SSA_GDP_data$Code)
copy_SSA_GDP_data$Country <- as.character(copy_SSA_GDP_data$Country)
copy_SSA_GDP_data$Region <- as.character(copy_SSA_GDP_data$Region)
```

✓ **Step 7**: Plot the correlation graph.

```
# Plotting the data set to evaluate for correlation

pairs.panels(SSA_findex_data[c(40:50)],cex=1)
```

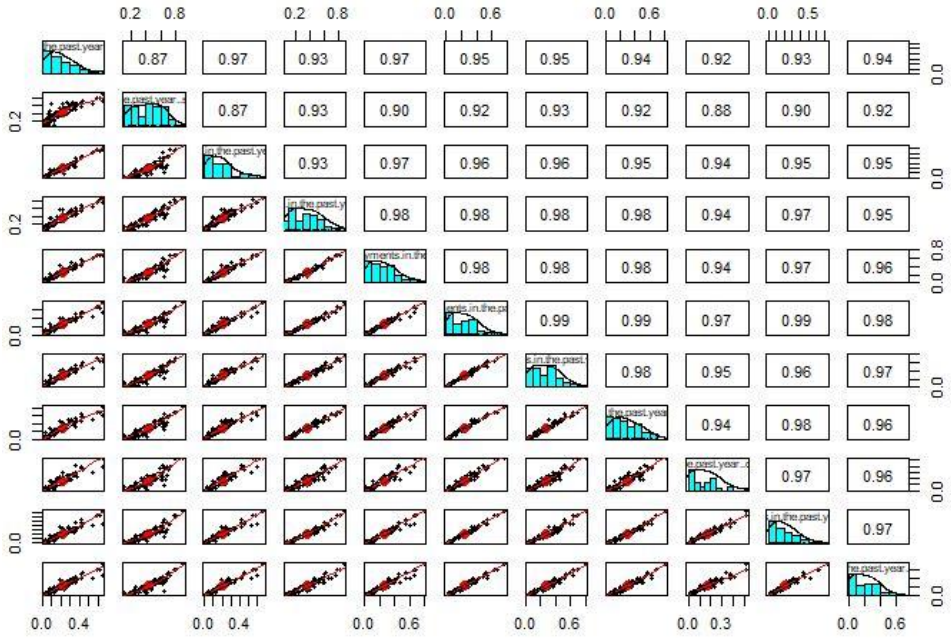

Figure 1: **Correlation Plot**

- ✓ **Step 8**: Merging the two data sources.

```
#  Merging the global Findex data and GDP data
new_merged_dataset <- merge (SSA_findex_data, copy_SSA_GDP_data, by =
c('Year','Code','Country','Region'))

copy_new_merged_dataset  <- new_merged_dataset
```

4 Data Cleaning

- ✓ **Step 9**: Check and impute missing values in the merged dataset. This was done using mean imputation.
- ✓ **Step 10**: Check for duplicates**.**
- ✓ **Step 11**: Feature encoding of categorical data to numeric data.

```
# Feature Encoding For The Income Level Column

str(new_merged_dataset$Income.Level)
sapply(new_merged_dataset,class)
income_model_matrix<-model.matrix(data = new_merged_dataset,GDP ~ Income.Level-1)
plot(new_merged_dataset)
View(income_model_matrix)
```

```
new_merged_dataset3 <- cbind(new_merged_dataset[,1:2],income_model_matrix)
new_merged_dataset <-cbind(new_merged_dataset3,new_merged_dataset[4:79])
colnames(new_merged_dataset)
```

5 Data Transformation

- ✓ **Step 12**: Examine for multicollinearity.

```
#  Examining multicollinearity using eigensystem analytics

cor_val<- eigen(cor(new_merged_dataset[,8:80]))$values

round(cor_val,5)
 [1] 47.48384 15.33252  2.90650  1.84874  1.21090  0.77562  0.74026
 [8]  0.44215  0.41574  0.32109  0.25808  0.19932  0.17451  0.15403
[15]  0.13051  0.08636  0.06933  0.06191  0.05873  0.04951  0.04372
[22]  0.03252  0.03178  0.02656  0.02385  0.01852  0.01727  0.01321
[29]  0.01183  0.00889  0.00779  0.00679  0.00597  0.00511  0.00432
[36]  0.00395  0.00323  0.00290  0.00217  0.00189  0.00179  0.00143
[43]  0.00127  0.00079  0.00074  0.00055  0.00050  0.00029  0.00026
[50]  0.00022  0.00010  0.00007  0.00003  0.00002  0.00000  0.00000
[57]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
[64]  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000  0.00000
[71]  0.00000  0.00000  0.00000

kappa(round(cor(new_merged_dataset[,50:60]),2))

[1] 444.9049
```

- ✓ **Step 13**: Normalize Dataset.

```
# Normalizing the data
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}

str(new_merged_dataset)
norm.dataset <-as.data.frame(apply(new_merged_dataset[7:80],2, normalize))
norm_data <- cbind(new_merged_dataset[1:6],norm.dataset)
```

- ✓ **Step 14**: Partitioning data into two sets of 70% for the training and 30% for the test data.

```
# Data Partitioning
```

```
set.seed(222)
ind <- sample(2, nrow(norm_data),replace = TRUE, prob = c(0.7,0.3))
traindata <-norm_data[ind==1,]
testdata <- norm_data[ind==2,]
str(testdata)
```

- ✓ **Step 15**: Regularize the train dataset to address multicollinearity.

```
# Regularization with elastic net
set.seed(1234)
en <- train(GDP ~. -Year -Country,
            traindata,
            method ='glmnet',
            tuneGrid =  expand.grid(alpha=seq(0,1,length=10),
                                    lambda = seq(0.0001,0.0,length=5)),
            trControl = custom)

# Evaluating regularization graphs
plot(en)
plot(en$finalModel,xvar = 'lambda',label=TRUE)
plot(en$finalModel,xvar = 'dev',label=TRUE)

# Plotting variable importance graph
var_train_imp <- varImp(en,scale = FALSE)
plot(var_train_imp,top=25)
```

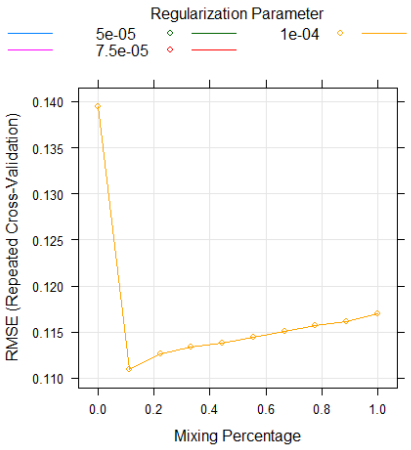

Figure 2: *Regularization Parameter from elastic net*

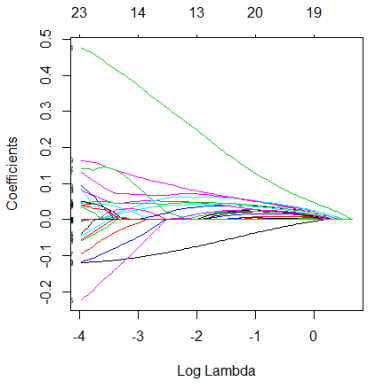

Figure 3: Coefficient *vs Log Lambda from elastic net*

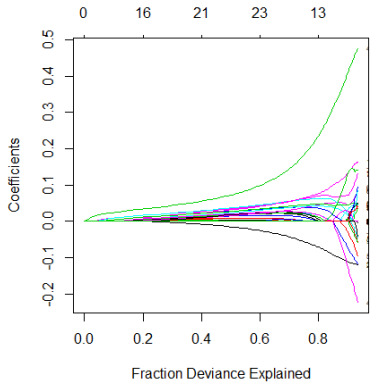

Figure 4: *Variable Coefficient vs Fraction deviance from elastic net*

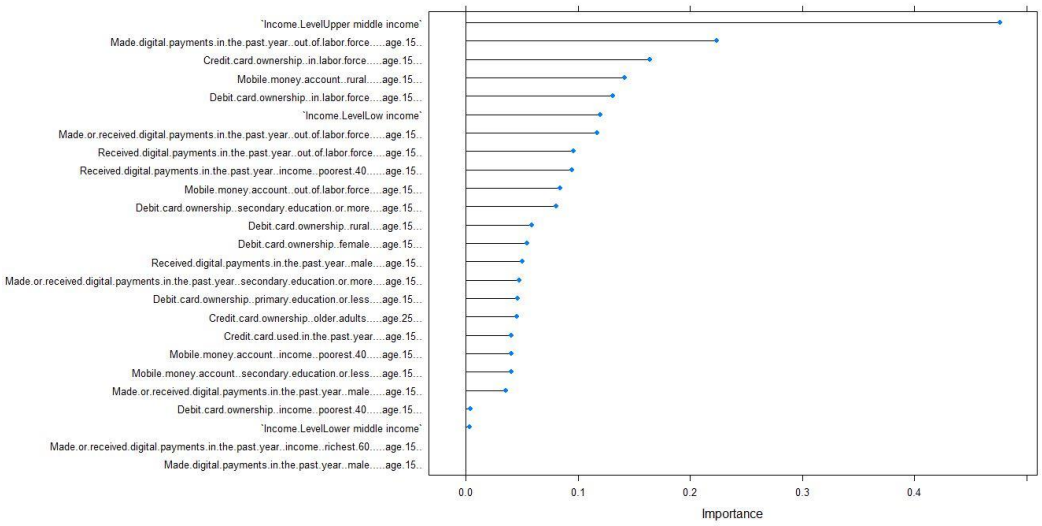

Figure 5*: Variable Importance Graph*

- ✓ **Step 16:** Predicting the regularized data to evaluate accuracy

```
#Prediction For Elastic Net Using The Test And Train Data Set

p1 <- predict(en,traindata)
sqrt(mean((traindata$GDP-p1)^2))
[1] 0.06091626

# For Test Data
p2 <- predict(en,testdata)
sqrt(mean((testdata$GDP-p2)^2))
[1] 0.09084775
```

Step 17: Create a new dataset from the variable importance for the regularization step

```
# Creating New Dataset With The Variables From Elastic Net

new_data <- norm data[,col_names,drop=FALSE]
copy_new_data<- new_data
View(new_data)
colnames(new_data)

# Data Partitioning of New Data
set.seed(222)
index <- sample(2, nrow(new_data),replace = TRUE, prob = c(0.70,0.30))
train.data <-new_data[index==1,]
test.data <- new_data[index==2,]
```

6 Data Mining & Analysis

- ✓ **Step 18**: Install relevant libraries: *caret, glmnet, psych, e1071, pls*.
- ✓ **Step 19**: Build the Partial Least Square Regression (PLSR) model with the train dataset.
- ✓ **Step 20**: Predict the dependent variables with the test data and validate the results by comparing the reliability values ($R^{2,}$ RMSE, MSE) with the test dataset.
- ✓ **Step 21**: Build the Support Vector Regression (SVR) model on the training data.
- ✓ **Step 22**: Predict the dependent variables with the test data and validate the results by comparing the reliability values ($R^{2,}$ RMSE, MSE) with the test dataset.
- ✓ **Step 23**: Build the LASSO regression model on the training data.
- ✓ **Step 24**: Predict the dependent variables with the test data and validate the results by comparing the reliability values ($R^{2,}$ RMSE, MSE) with the test dataset.
- ✓ **Step 25**: Build the Principal Component Regression (PCR) model on the training data.
- ✓ **Step 26**: Predict the dependent variables with the test data and validate the results by comparing the reliability values ($R^{2,}$ RMSE, MSE) with the test dataset.

7 Evaluation

- ✓ **Step 27:** Compare all the reliability values from the models SVR, PLSR, PCR and LASSO.
- ✓ **Step 28:** The best performing model to be selected should be the model with the highest R^2 should be picked as the best model.
- ✓ **Step 29:** Evaluate the validation plot for the different models.

```
# Partial Least Regression

set.seed(1234)
library(pls)
pls_model2 <- plsr(GDP~.,data= train.data,ncomp=10,validation='CV')
summary(pls_model2)
```

```
RMSEP(pls_model2)

# Predicting for train data with pls model
pls_predict<- predict(pls_model2,train.data,ncomp=7)

RMSE_pls1 <- RMSE(pls_predict, train.data$GDP)
MSE_pls1 <-mean((train.data$GDP-pls_predict)^2)
R2_pls1<-1-(sum((train.data$GDP-pls_predict)^2)/sum((train.data$GDP-
mean(train.data$GDP))^2))

# Predicting  For Test Data
pls_predict2<- predict(pls_model2,test.data,ncomp =7)

# Evaluating the model
RMSE_pls <- RMSE(pls_predict2, test.data$GDP)
MSE_pls <-mean((test.data$GDP-pls_predict2)^2)
R2_pls <- 1-(sum((test.data$GDP-pls_predict2)^2)/sum((test.data$GDP-
mean(test.data$GDP))^2))

cat("\n", "MSE:", MSE_pls, "\n",
   "RMSE:", RMSE_pls, "\n", "R-squared:", R2_pls)
```

MSE: 0.002382553
RMSE: 0.0488114
R-squared: 0.976749

```
# Principal Componet Regression
set.seed(1234)
pcr_model <- pcr(GDP~., data=train.data, validation='CV')
summary(pcr_model)


plot(pcr_model, "validation", val.type = "R2",
    main = "R squared for PCR Model",
    xlab ='No of components',
    ylab ='R^2')

validationplot(pcr_model,val.type = 'MSEP')
plot(pcr_model,'validation',val.type='MSEP',
    main='Error rates for the PCR Model')

# PCR Prediction For Train Data
pcr.predict1  <- pcr_model %>% predict(train.data,ncomp=12)

RMSE_pcr <- sqrt(mean((train.data$GDP-pcr.predict1)^2))
mse_pcr <- mean((train.data$GDP-pcr.predict1)^2)
r2_pcr <- 1-(sum((train.data$GDP-pcr.predict1)^2)/sum((train.data$GDP-
mean(train.data$GDP))^2))
```

```
cat(" MAE:", mae_pcr, "\n", "MSE:", mse_pcr, "\n",
    "RMSE:", RMSE_pcr, "\n", "R-squared:", r2_pcr)

# Pcr Prediction For Test Data
pcr.predict  <- pcr_model %>% predict(test.data,ncomp=12)
summary(pcr.predict)

RMSE_pcr2 <- sqrt(mean((test.data$GDP-pcr.predict)^2))
mse_pcr2 <- mean((test.data$GDP-pcr.predict)^2)
mae_pcr2 <-mean(abs(test.data$GDP-pcr.predict))
r2_pcr2 <- 1-(sum((test.data$GDP-pcr.predict)^2)/sum((test.data$GDP-
mean(test.data$GDP))^2))

cat(" MAE:", mae_pcr2, "\n", "MSE:", mse_pcr2, "\n",
    "RMSE:", RMSE_pcr2, "\n", "R-squared:", r2_pcr2)
```

MAE: 0.05291646
MSE: 0.006606115
RMSE: 0.08127801
R-squared: 0.9355319

```
# Support Vector Regression
svm_model_linear <- svm(GDP~.,data=train.data,type = 'eps-regression',
kernel='linear',cost=1.0,epsilon=0.1,validation='CV')
svm_predict_train1<- predict(svm_model_linear,data=train.data$GDP)

# Evaluating the model linear
svm_RMSE_linear <- RMSE(svm_predict_train1, train.data$GDP)
MSE_svm_linear <-mean((train.data$GDP-svm_predict_train1)^2)
mae_svm_linear <- MAE(svm_predict_train1, train.data$GDP)
R2_svm_linear <- 1-(sum((train.data$GDP-svm_predict_train1)^2)/sum((train.data$GDP-
mean(train.data$GDP))^2))

##Prediction for test data
svm_predict_linear<- predict(svm_model_linear,data=test.data$GDP)

svm_RMSE_linear2 <- RMSE(svm_predict_linear, test.data$GDP)
MSE_svm_linear2 <-mean((test.data$GDP-svm_predict_linear)^2)
R2_svm_linear2 <- 1-(sum((test.data$GDP-svm_predict_linear)^2)/sum((test.data$GDP-
mean(test.data$GDP))^2))

cat(" MAE:", mae_svm2, "\n", "MSE:", MSE_svm2, "\n",
    "RMSE:", svm_RMSE2, "\n", "R-squared:", R2_svm_linear2)
```

MAE: 0.021648
MSE: 0.0004943634
RMSE: 0.02223428
R-squared: 0.9913184

```
#Lasso Regression
```

```
x_vars <- model.matrix(GDP~. , new_data)[,-1]
y_var <- new_data$GDP
lambda_seq <- 10^seq(2, -2, by = -.1)

set.seed(1234)
train = sample(1:nrow(x_vars), nrow(x_vars)/2)
x_test = (-train)
y_test <- y_var[x_test]

cv_lasso <- cv.glmnet(x_vars[train,], y_var[train], alpha=1,
                                        lambda =lambda_seq,
                  nfolds = 10)
# identifying best lamda
best_lam <- cv_lasso$lambda.min
best_lam

# Rebuilding the model with best lamda value identified
lasso_best <- glmnet(x_vars[train,], y_var[train], alpha = 1, lambda = best_lam)
pred <- predict(lasso_best, s = best_lam, newx = x_vars[x_test,])

# getting coefficients
out<- glmnet(x_vars[train,], y_var[train],alpha = 1,lambda =lambda_seq)
lasso.coef<-predict(out,type='coefficients',s=best_lam)
lasso.coef
lasso.coef[lasso.coef!=0]

# Calculating Metrics Statistics For R
actual <- y_var[x_test]
preds <- pred
rss <- sum((preds - actual) ^ 2)
tss <- sum((actual - mean(actual)) ^ 2)
rsq <- 1 - rss/tss
rsq
[1] 0.8441346

MSE_lasso <-mean((y_var[x_test]-pred)^2)
[1] 0.01051933

RMSE_lasso <- RMSE(pred,y_var[x_test])
[1] 0.1025638

MSE_lasso2 <-mean((y_var[x_test]-pred)^2)
[1] 0.01051933

RMSE_lasso2 <- RMSE(pred,y_var[x_test])
[1] 0.1025638
```