Configuration Manual

MSc Research Project
FinTech

Ifeoma Njoh-Paul
Student ID: X18199721

School of Computing
National College of Ireland

Supervisor: Mr Victor Del Rosal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Ifeoma Njoh-Paul		
Student ID:	X18199721		
Programme	Msc FinTech	**Year:**	2019/2020
Module:	Research Project		
Lecturer:	Mr Victor Del Rosal		
Submission Due Date:	17 Aug 2020		
Project Title:	A Comparative Study of Ensemble Techniques and Individual Classifiers in Predicting Insurance Claim		
Word Count:	1,409	**Page Count:**	12

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:	Ifeoma Njoh-Paul
Date:	17 Aug 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

Ifeoma Marian Njoh-Paul
Student ID: x18199721

1. Introduction

This configuration manual aims to provide the necessary technical resources and step by step process used to conduct the research project named "*A comparative study of ensemble techniques and individual classifiers in predicting insurance claim".*

2. System Requirements

2.1Hardware

- HDD: 500GB
- System type: 64-bit operating system, x64-based processor
- Processor: Intel(R) Core(TM) i7-4600U CPU @ 2.10GHz 2.70 GHz
- RAM: 8GB

2.2Software

- R studio: This was used for the statistical analysis and machine learning process, the version used was 3.5.3
- Microsoft word: This was used for writing the research report

3. Data Extraction and Cleaning

- **Step 1:** The first step was to download the data from the Kaggle website[1]
- **Step 2:** The next step was to import the data into the R studio for the analysis process
- **Step 3:** Check the structure of the data and also the number of rows and columns to get more insights

```
#Loading the dataset
dat = read.csv("train.csv")
#view dataset
names(dat)
nrow(dat)
ncol(dat)
```

[1] https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data?select=train.csv

str(dat)
There are 595212 rows and 56 columns/variables in the dataset as seen in fig 1, out of the 56 variables, 30 are categorical and 20 are continuous.

```
> nrow(dat)
[1] 595212
> ncol(dat)
[1] 56
```

Figure 1: The rows and columns of the dataset

- **Step 4:** Checking for missing values and imputation. According to the data description, all missing values are represented by -1. This research changed all -1 values to NA's for the imputation process.

```
######## Changing -1's to NA's ##########

dat[dat == -1] <- NA
sum(is.na(dat))
sum(complete.cases(dat))  #rows with complete  observations, no NAs
```

```
> dat[dat == -1] <- NA
> sum(is.na(dat))
[1] 846458
```

Figure 2: Missing values in the dataset

- **Step 5:** The dataset is seen to have missing values of 846,458. The next step is the imputation process, the research used mode for categorical variables and mean for continuous variables. This step eliminated all missing values as seen in fig 3.

```
###imputing missing values using mean and mode from ImputeMissings package###
dat$ps_reg_03 = impute(dat$ps_reg_03, object = NA, method = "mean", flag = FALSE)
dat$ps_car_12 = impute(dat$ps_car_12, object = NA, method = "mean", flag = FALSE)
dat$ps_car_14 = impute(dat$ps_car_14, object = NA, method = "mean", flag = FALSE)
dat$ps_ind_02_cat = impute(dat$ps_ind_02_cat, object = NA, method = "mode", flag = FALSE)
dat$ps_ind_04_cat = impute(dat$ps_ind_04_cat, object = NA, method = "mode", flag = FALSE)
dat$ps_ind_05_cat = impute(dat$ps_ind_05_cat, object = NA, method = "mode", flag = FALSE)
dat$ps_car_01_cat = impute(dat$ps_car_01_cat, object = NA, method = "mode", flag = FALSE)
dat$ps_car_02_cat = impute(dat$ps_car_02_cat, object = NA, method = "mode", flag = FALSE)
dat$ps_car_07_cat = impute(dat$ps_car_07_cat, object = NA, method = "mode", flag = FALSE)
dat$ps_car_09_cat = impute(dat$ps_car_09_cat, object = NA, method = "mode", flag = FALSE)
dat$ps_car_11= impute(dat$ps_car_11, object = NA, method = "mode", flag = FALSE)
```

```
> sum(is.na(dat))
[1] 0
```

Fig 3: No missing values after imputation

4. Data Transformation

- **Step 1:** A random sampling was done to reduce the size of the dataset for easy computation
- **Step 2:** Variable selection was conducted

###Fitting a regression model####

```
rPartMod <- train(target ~ ., data=newdata, method="glm")
rrfImp <- varImp(rPartMod, scale=F)
rrfImp
plot(rrfImp, top = 20, main='Variable Importance')
```

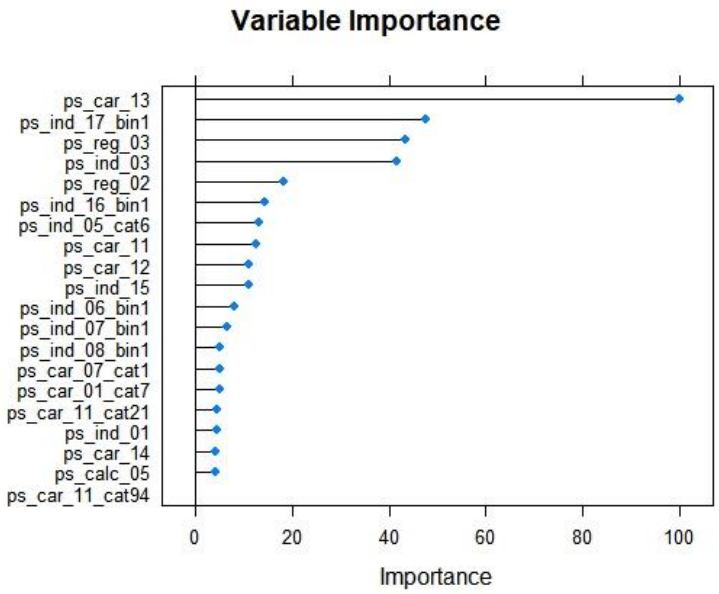

Figure 4: Variable importance through Gbm

- **Step 3:** Normalize the dataset

####Normalize the dataset#######

```
normalization <- function(x){
  return ((x - min(x)) / (max(x) - min(x)))
}
newdata1= as.data.frame(sapply(newdata1[, -1], normalization))
Data2 = cbind(newdata1, newdata[, 1])
newdata1<-Data2
colnames(newdata1)[21]<-"target"
```

- **Step 4:** The training dataset was balanced by a hybrid approach using the Caret package as seen in Fig 5

```
#Partition the data into training and testing #
split <- createDataPartition(newdata1$target, p=0.7, list=FALSE)
train <- newdata1[split,]
test <- newdata1[-split,]
plot(newdata1$target)
table(newdata1$target)

#Balance the train data#
BalTraining <-ovun.sample(target~.,data=train,method = "both",N=97000)$data
plot(BalTraining$target)
```

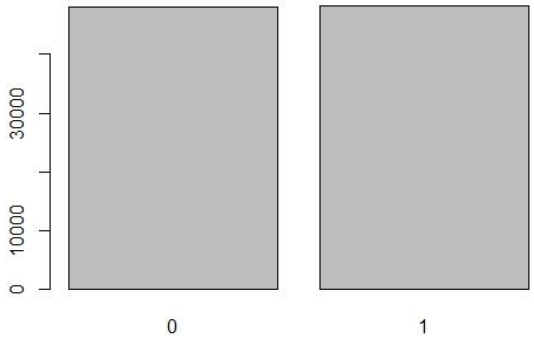

Figure 5: Balanced dataset

5. Data Mining

- **Step 1:** Compute the Support Vector Machine (SVM) on the training data
- **Step 2:** Confirm the results using the test dataset
- **Step 3:** Compute the confusion matrix
- **Step 4:** Plot the ROC/AUC curve

Table 1: SVM computation and result

###SVM### SVMmodel<-svm(target~.,data=BalTraining) pred <- predict(SVMmodel, test) confusionMatrix(data = pred, test$target, positive= "1") **#####Plotting ROC curve#####** ROCpred<-roc.curve(test$target, pred,main="Support Vector Machine") legend('bottomright','AUC-0.541',lty = 5,bty = 'n',inset = c(0,0.50))	Accuracy : 0.7394 95% CI : (0.7344, 0.7443) No Information Rate : 0.9639 P-Value [Acc > NIR] : 1 Kappa : 0.0212 Mcnemar's Test P-Value : <2e-16 Sensitivity : 0.75482 Specificity : 0.32625 Pos Pred Value : 0.96768 Neg Pred Value : 0.04743 Prevalence : 0.96393 Detection Rate : 0.72759 Detection Prevalence : 0.75189 Balanced Accuracy : 0.54053 'Positive' Class : 1

- **Step 5:** Compute the Logistic regression (GLM) on the training data
- **Step 6:** Confirm the results using the test dataset
- **Step 7:** Compute the confusion matrix
- **Step 8:** Plot the ROC/AUC curve

Table 2: GLM computation and result

###Logistic regression### GlmModel<-train(target~.,data=BalTraining,method="glm", family="binomial") Glmpred <- predict(GlmModel, test) confusionMatrix(data = Glmpred, test$target, positive= "1") **#####Plotting ROC curve#####** ROCGlm<-roc.curve(test$target, Glmpred,main="Logistic regression") legend('bottomright','AUC-0.573',lty = 5,bty = 'n',inset = c(0,0.50))	Accuracy : 0.6287 95% CI : (0.6232, 0.6341) No Information Rate : 0.9639 P-Value [Acc > NIR] : 1 Kappa : 0.0268 Mcnemar's Test P-Value : <2e-16 Sensitivity : 0.63295 Specificity : 0.51386 Pos Pred Value : 0.97206 Neg Pred Value : 0.04978 Prevalence : 0.96393 Detection Rate : 0.61012 Detection Prevalence : 0.62765 Balanced Accuracy : 0.57341 'Positive' Class : 1

- **Step 9:** Compute the Artificial Neural Network on the training data
- **Step 10:** Confirm the results using the test dataset
- **Step 11:** Compute the confusion matrix
- **Step 12:** Plot the ROC/AUC curve

Table 3: ANN computation and result

###Artificial Neural Network### allVars <- colnames(BalTraining) predictorVars <- allVars[!allVars%in%'target'] predictorVars <- paste(predictorVars, collapse = "+") f <- as.formula(paste("target~",predictorVars, collapse = "+")) NN1<-neuralnet(formula= f,data=BalTraining,hidden=1,linear.output=F, stepmax = 1e5, threshold = 0.2,rep = 1) pLOTNN <- plot(NN1) **###Predicting for ANN###** predictedtest = compute(NN1, test[,c(1:20)]) predict_testNN= (predictedtest$net.result) predictedresults <- data.frame(actual = test$target, prediction = predict_testNN[,2]) predictedresults roundresult<-sapply(results[,2],round,digits=0) roundresult=as.factor(roundresult) roundresultdf=data.frame(roundresult) attach(roundresultdf) Anntab<-table(actual = test$target,prediction=roundresult) Anntab confusionMatrix(Anntab, positive= "1") **####ROC curve for Neural network#####** ROCAnn<-roc.curve(test$target, roundedresults,main="Artificial neural network") legend('bottomright','AUC-0.571',lty = 5,bty = 'n',inset = c(0,0.50))	Accuracy : 0.5996 95% CI : (0.594, 0.6051) No Information Rate : 0.5967 P-Value [Acc > NIR] : 0.1571 Kappa : 0.024 Mcnemar's Test P-Value : <2e-16 Sensitivity : 0.97218 Specificity : 0.04827 Pos Pred Value : 0.60179 Neg Pred Value : 0.53974 Prevalence : 0.59669 Detection Rate : 0.58009 Detection Prevalence : 0.96393 Balanced Accuracy : 0.51022 'Positive' Class : 1

- **Step 13:** Compute the Linear Discriminate Analysis (LDA) on the training data
- **Step 14:** Confirm the results using the test dataset
- **Step 15:** Compute the confusion matrix
- **Step 16:** Plot the ROC/AUC curve

Table 4: LDA computation and result

```
###Linear Discriminate Analysis###
Ldamodel<-lda(target~.,data=BalTraining)
predlda <- predict(Ldamodel, test)
tab<-table(data = predlda$class, test$target)
confusionMatrix(tab, positive= "1")

####ROC curve for LDA###
ROCALda<-roc.curve(test$target,
predlda$class, main="Linear Descriminate
Analysis")
legend('bottomright','AUC-0.574',lty = 5,bty =
'n',inset = c(0,0.50))
```

```
Accuracy : 0.6312
95% CI : (0.6257, 0.6366)
No Information Rate : 0.9639
P-Value [Acc > NIR] : 1

Kappa : 0.0271

Mcnemar's Test P-Value : <2e-16

Sensitivity : 0.63561
Specificity : 0.51201
Pos Pred Value : 0.97208
Neg Pred Value : 0.04995
Prevalence : 0.96393
Detection Rate : 0.61269
Detection Prevalence : 0.63029
Balanced Accuracy : 0.57381

'Positive' Class : 1
```

- **Step 17:** Compute the Extreme Gradient Boosting (XGBOOST) on the training data
- **Step 18**: Confirm the results using the test dataset
- **Step 19:** Compute the confusion matrix

Table 5: XGBOOST computation and result

```
######Creating matrix #####
trainm<-sparse.model.matrix(target~.,-1,data=train1)
head(trainm)
train_label<-train1[,"target"]
train_label<-as.numeric(train_label)-1
train_matrix<-xgb.DMatrix(data=
as.matrix(trainm),label=train_label)
testm<-sparse.model.matrix(target~.,-1,data=test1)
head(testm)
test_label<-test1[,"target"]
test_label<-as.numeric(test_label)-1
test_matrix<-xgb.DMatrix(data=
as.matrix(testm),label=test_label)

######Setting the parameters####
nc<-length(unique(train_label))
xgb_param<-list("objective"="multi:softprob",
            "eval_metric"="mlogloss",
             booster="gbtree",
            "num_class"=nc)

#####Building the model#####
XGmod<-xgb.train(params = xgb_param,
```

```
Accuracy : 0.9639
95% CI : (0.9617, 0.966)
No Information Rate : 0.9639
P-Value [Acc > NIR] : 0.5204

Kappa : -1e-04

Mcnemar's Test P-Value : <2e-
16

Sensitivity : 1.0000
Specificity : 0.0000
Pos Pred Value : 0.9639
Neg Pred Value : 0.0000
Prevalence : 0.9639
Detection Rate : 0.9639
Detection Prevalence : 1.0000
Balanced Accuracy : 0.5000

'Positive' Class : 1
```

data = train_matrix, nrounds =15 , watchlist = watchlist, gamma=0) **#####Prediction of the model####** Xgpred<-predict(XGmod,newdata = test_matrix) confusionMatrix (Xgpred1, test_label) Xpred<-matrix(Xgpred,nrow = nc,ncol = length(Xgpred)/nc) %>% t() %>% data.frame()%>% mutate(label= test_label,max_prob=max.col(.,"last")-1) **####Accuracy level###** Xgtab<- table(prediction=Xpred$max_prob,Actual=Xpred$label) confusionMatrix(Xgtab, positive= "1")	

- **Step 20**: Compute the Stacking on the training data
- **Step 21:** Confirm the results using the testing and validation dataset
- **Step 22:** Compute the confusion matrix

Table 6: Stacking model computation and result

#####Stacking Method##### **#Split the dataset into training, testing and validation##** Training <- 0.70 Validation <- 0.15 Testing <- 0.15 sample_Training <- floor(Training * nrow(BalTraining)) sample_Validation <- floor(Validation * nrow(BalTraining)) sample_Test <- floor(Testing * nrow(BalTraining)) indiceTraining <- sort(sample(seq_len(nrow(BalTraining)), size=sample_Training)) indiceNotTraining <- setdiff(seq_len(nrow(BalTraining)), indiceTraining)	Accuracy : 0.7526 95% CI : (0.7455, 0.7596) No Information Rate : 0.5025 P-Value [Acc > NIR] : <2e-16 Kappa : 0.5051 Mcnemar's Test P-Value : 0.9336 Sensitivity : 0.7542 Specificity : 0.7509 Pos Pred Value : 0.7536 Neg Pred Value : 0.7515 Prevalence : 0.5025 Detection Rate : 0.3790 Detection Prevalence : 0.5030 Balanced Accuracy : 0.7526 'Positive' Class : 1

```
indiceValidation  <-
sort(sample(indiceNotTraining,
size=sample_Validation))
indiceTest        <- setdiff(indiceNotTraining,
indiceValidation)

dfTraining   <- BalTraining[indiceTraining, ]
dfValidation <- BalTraining[indiceValidation, ]
dfTest       <- BalTraining[indiceTest, ]
View(dfTest)

###Fit the models###
modelFitSVM <- svm(target ~., data =
dfTraining)
modelFitGlm<-
train(target~.,data=dfTraining,method="glm",
family="binomial")
modelFitLda<- train(target ~., data =
dfTraining, method='lda')

##build the artificial neural network
model##
allVars1 <- colnames(dfTraining)
predictorVars1 <-
allVars1[!allVars1%in%'target']
predictorVars1 <- paste(predictorVars1,
collapse = "+")
f1 <-
as.formula(paste("target~",predictorVars1,
collapse = "+"))
NN2<-neuralnet(formula=
f1,data=dfTraining,hidden=1,linear.output=F,
stepmax = 1e5, threshold = 0.1,rep = 1 )

##validate using the validation data##
predGlm <- predict(modelFitGlm, newdata =
dfValidation)
predsvm <- predict(modelFitSVM, newdata =
dfValidation)
predlda <- predict(modelFitLda, newdata =
dfValidation)

prednn <- compute(NN2,
dfValidation[,c(1:21)])
predict_dftestNN = (predict_testNN$net.result)

##Combine all the models in a dataframe##
predDF <- data.frame(predGlm, predsvm,
predlda, target = dfValidation$target,
stringsAsFactors = F)
```

##Build the stack model## modelStack <- train(target ~ ., data = predDF, method = "glm") **##Predict using the testing data##** testPredGlm <- predict(modelFitGlm, newdata = dfTest) testPredsvm <- predict(modelFitSVM, newdata =dfTest) testPredlda <- predict(modelFitLda, newdata = dfTest) **##Combine the testing results in a dataframe##** testPredLevelOne <- data.frame(testPredGlm, testPredsvm, testPredlda, target = dfTest$target, stringsAsFactors = F) combPred <- predict(modelStack, testPredLevelOne) **##Confusion matrix for stacking##** confusionMatrix(combPred, dfTest$target, positive = "1")	

6. Evaluation

Step 1: Compare the results of the ROC/AUC four individual results

####ROC Curves of the different models#####

Roclist<-roc.curve(test$target,predlda$class ,plot = TRUE, xlab="False positive percentage", ylab= "True positive percentage",col="blue",main="The individual classifiers ROC comparison graph")
roc.curve(test$target,pred,col="green",add=TRUE)
roc.curve(test$target, roundedresults,col="red",add=TRUE)
roc.curve(test$target, Glmpred,col="black",add=TRUE)
legend("bottomright",legend=c("LDA","SVM","GLM","ANN"),col = c("blue","green","red","black"),lwd=4)

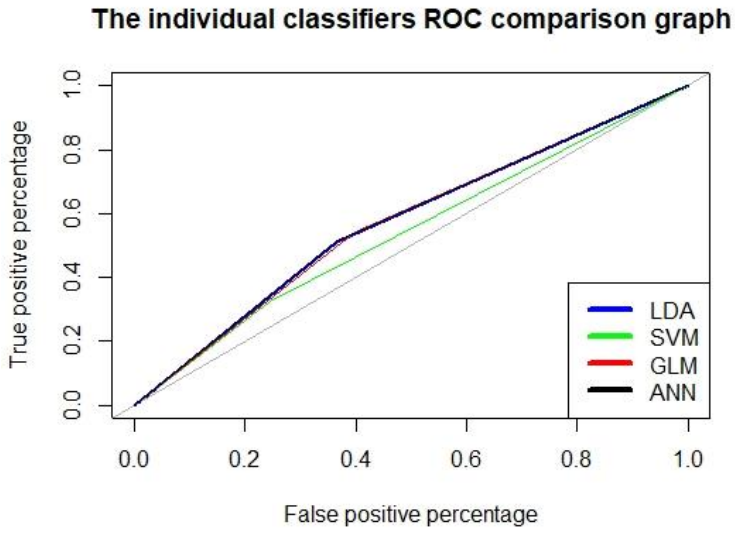

Figure 6: ROC comparison

Step 2: Compare the selected performance measures of the various models

```
#plotting performance measures comparison##

Dataset<-matrix(c(73.94, 75.48, 54.1, 62.87, 63.3, 57.3, 60, 80.2, 57.1, 63.1, 63.6, 57.4, 96, 99, 52, 75.26, 75.42, 76.05 ), ncol = 6, byrow = F)
rownames(Dataset)<-c("Accuracy", "Sensitivity", "AUC")
colnames(Dataset)<-c("SVM", "GLM", "ANN","LDA", "XGBOOST", "Stacking")
Dataset<-as.table(Dataset)
Dataset

colour.names=c('blue','brown', 'purple')
barplot(Dataset, legend=T, beside = T, main = "Performance measures", ylim = c(0, 150), col = colour.names, args.legend = list(x = ncol(Dataset) + 17, bty = "n"))
```

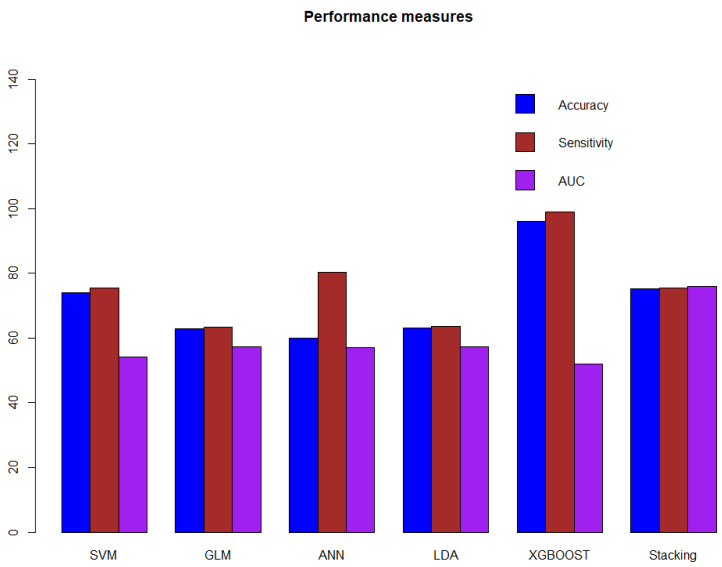

Figure 7: Performance measures of the models

7. References

1. https://www.kaggle.com/c/porto-seguro-safe-driver-prediction/data?select=train.csv
2. https://rstudio.com/products/rstudio/download/