Exam 2020 Project Report

Yuan Du

SECTION 1: EXECUTIVE SUMMARY

The goal of this project is to develop a model that can correctly predict the state of the binary target. One training dataset "EXAM_TRAINData.CSV" was provided for training. And the test dataset "EXAM_TESTData.CSV" including the same information except the target status was provided for prediction.

High level summary of all major findings: In the project, I developed not only a model to predict the probabilities of ID's status of target but also built another model to predict the probabilities of the target state. I summarize two models in the followings.

In the first model, the purpose is to predict the state of the target variable "TARGET". I used the variable "wheezy_copper_turtle_magic" as the binning class variable and divided the data into 512 subsets and modeled each subset. To build the model, I selected important variables by standard deviation over 1.5 as the predictors to compare the modeling performances with different approaches like Logistic Regression, Lasso and QDA. I found the QDA method was the best one. The model performance was excellent and has an average AUC of ROC curve with 96%. The most important ten factors are cheeky_plum_fox_noise,hasty_ivory_dragonfly_goose, gloppy_cerise_snail_contributor,flimsy_turquoise_fox_kernel,dorky_purple_kiwi_hint,wheezy_red_iguana_entropy,messy_mauve_wolverine_ordinal,lousy_smalt_pinscher_dummy,muggy_pumpkin_scorpion_grandmast, gloppy_turquoise_quoll_goose. In the last, I predicted the test data with this case and obtained the probabilities of IDs' target status.

In the second model, the purpose is to predict the probability of each ID's target status. I used the same variables selection method by standard deviation in the training model as predictors. The data processing is the same as that of the first model. In the modeling, I build 512 models with selected variables as our predictors with the best approach QDA. The predictions of the probabilities of IDs' target status for the test data were reported.

The models can detect the important factors for the target. The following report summarizes data analysis and predictive modeling methods conducted to fulfill the objective of the project.

SECTION 2: Introduction Section

2.1 Datasets

The two datasets are summarized in Table 1. The training data set has one "ID" variable, one "TARGET" variable, and 256 numerical predictors (255 continuous and one integer/count). Each variable was consisted of four words. It is hard to figure out the logical meaning of each variable and what the predication is about. The only integer variable "wheezy_copper_turtle_magic" has 512 levels, ranges from 0 to 511. Similar variables are in the testing data set except the testing data set does not have the target variable.

Table 1: Data Structure

	Dataset Name	Obs (n)	Variable number (p)	Variable types (n)
Training	EXAM_TRAINData.CSV	235930	258	ID(1), Continuous (255), Count(1), Target(1)
Testing	EXAM_TESTData.CSV	26214	257	ID(1), Continuous (255), Count(1)

2.2 Problem Statements

In this exam, the biggest data analysis question is how to use "binning technique" to bin numerical predictors. The problem is that each numerical variable is non-normal distributed and has very high kurtosis. And the only integer variable "wheezy_copper_turtle_magic" has 512 levels, ranges from 0 to 511. In each level, the number of rows is around 500. The structure and the distribution of the data will have a big impact on the target prediction.

SECTION 3: Body Section

3.1 Data exploration

The first step was to explore the two datasets. Here are some examples of figures of the target, distributions of first four variables and variable "wheezy_copper_turtle_magic" in Figure 1 – Figure 4. Additional figures and the table of statistics that contains the following statistics for each numerical predictor: "Variable Name", "Mean", "Median", "Lower Quartile", "Upper Quartile", "standard deviation", "Skewness", "Kurtosis", and "Percentage of Observations outside three standard deviation" are in appendix and/or in the submission.

Target Variable: the target is binary (1/0). Figure 1 shows the frequency and percentage of the target status. As we can see, the target percentage of 1(n=118018) and 0(n=117912) is balanced at 50%.

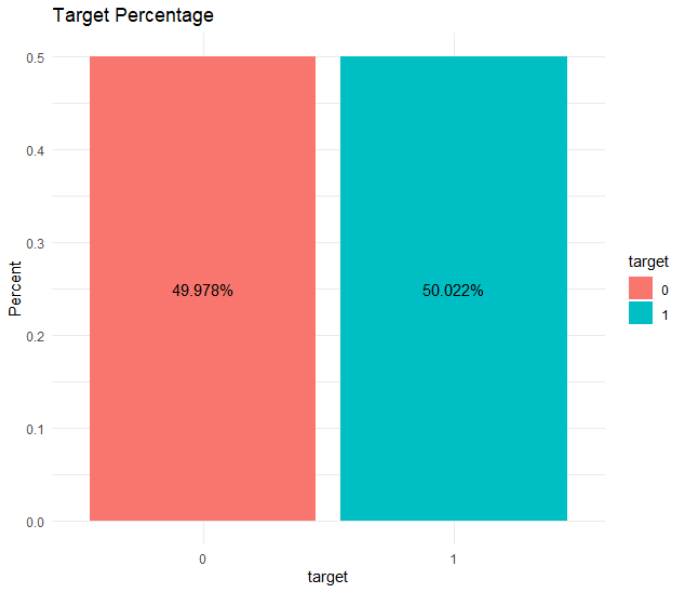
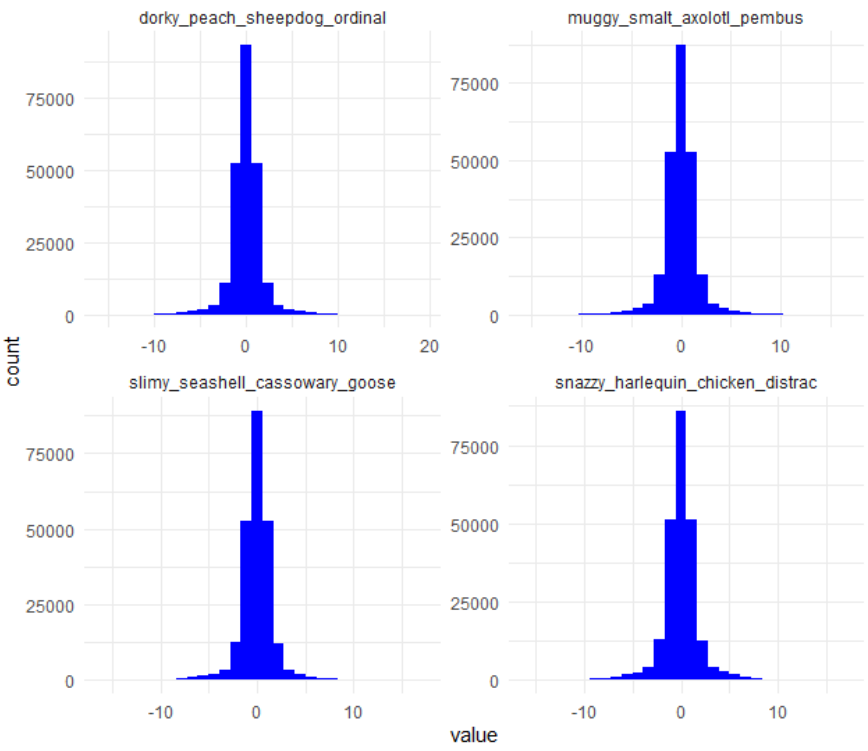
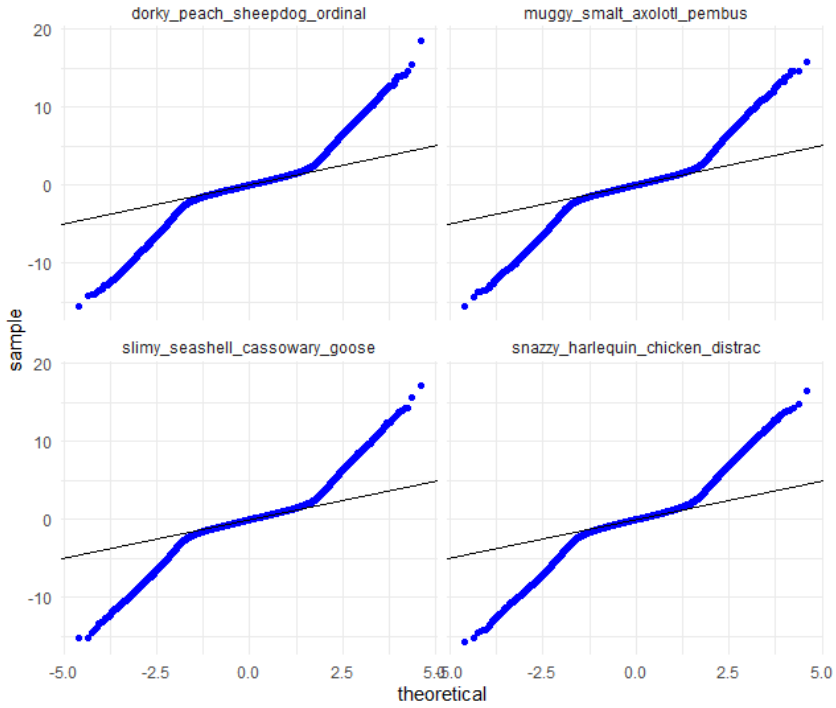


Figure 1: Target variable: target 1/0 Percent (%)

Figure 2 (a) and (b) show four examples of the distributions of first four variables and Q-Q plots. All variables have narrow shapes and peak in the middle, indicating high kurtosis and not normal distributed. Higher kurtosis corresponds to greater extremity of deviations (or outliers), and not the configuration of data near the mean.



(a)



(b)

Figure 2: First four variables (a) Distributions; (b) Q-Q plots

Figure 3 shows the bar charts on the variable “wheezy_copper_turtle_magic” with 512 levels. Each level is almost uniformly distributed with around 500 rows. This is also confirmed for the test dataset.

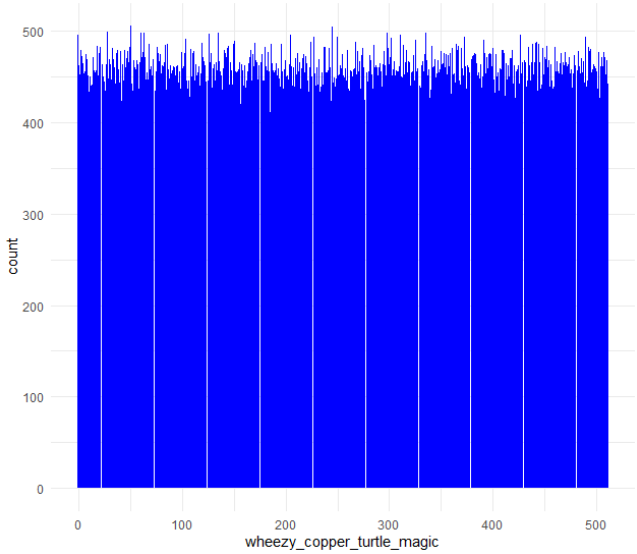
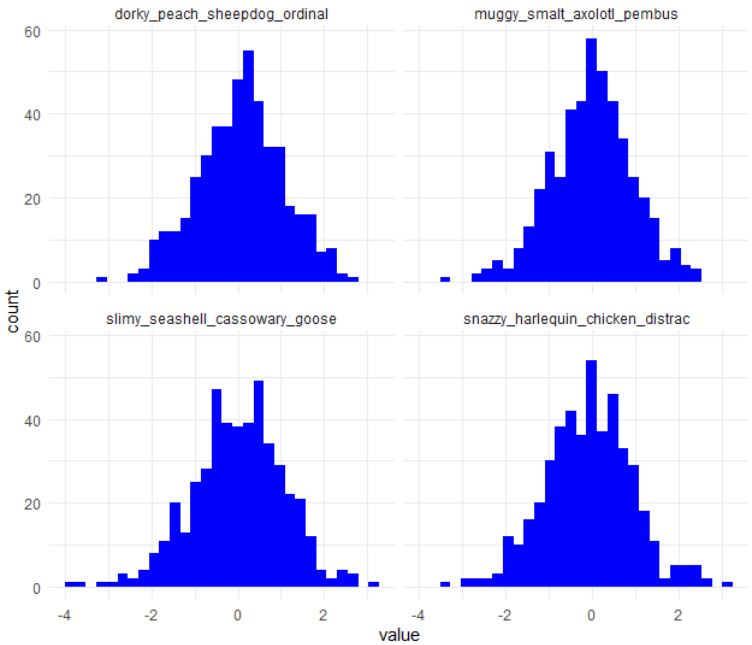
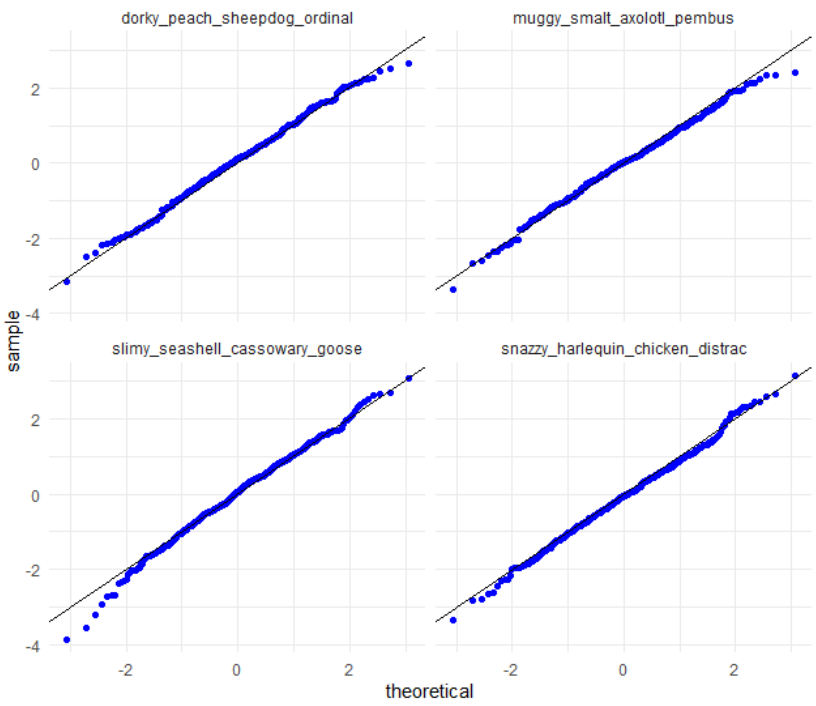


Figure 3: bar chart of variable “wheezy_copper_turtle_magic”

Figure 4 shows four examples of distributions of first four variables and Q-Q plots after dividing data into 512 subsets based on the variable “wheezy_copper_turtle_magic” with 512 levels. We can see that the distributions of variables are normal distributed within each subset.



(a)



(b)

Figure 4: First four variables after binning (a) Distributions; (b) QQ-plots

3.2 Data Preparation

After data exploration, I hypothesize that separate learning models for each train subset would provide a significantly better fit than the single model constructed for the entire train dataset. The next step is to divide data into 512 subsets for train and test dataset. This works like a magic because after the division/binning, each data subset is around 500 rows. And each variable is approximately normal distributed within each subset shown in Figure 4. This is important to answer our analysis questions. For this step, an example of the structure for three data subsets is shown in Figure 5. The final data is 512 train subsets and 512 test subsets.

	Column1	Column2	Column3	wheezy_copper_turtle_magic	Target	
row one	-0.8375	1.23875	0.62624	0	1	**Model 1**
row two	-1.274	0.35702	-0.2959	0	0	
row three	-0.4514	1.94546	3.06329	0	1	
	Data1-1	Data1-2	Data1-3			
row four	-0.2165	1.56234	0.16813	1	0	**Model 2**
row five	-0.1942	4.0494	0.90178	1	1	
row six	-0.4801	0.29027	-1.1621	1	1	
	Data2-1	Data2-2	Data2-3			
row seven	0.86997	0.78336	1.41082	2	1	**Model 3**
row eight	0.79237	-0.4544	2.26926	2	1	
row nine	-1.4229	-2.0213	0.29857	2	0	
	Data3-1	Data3-2	Data3-3			

Figure 5: Data division/binning example

Figure 6 shows first two variables' distributions by target status before (all train data) and after binning/division in data subset 1. We can see that after the division, the variables' variations and mean differences on target 1 vs 0 is much more obvious.

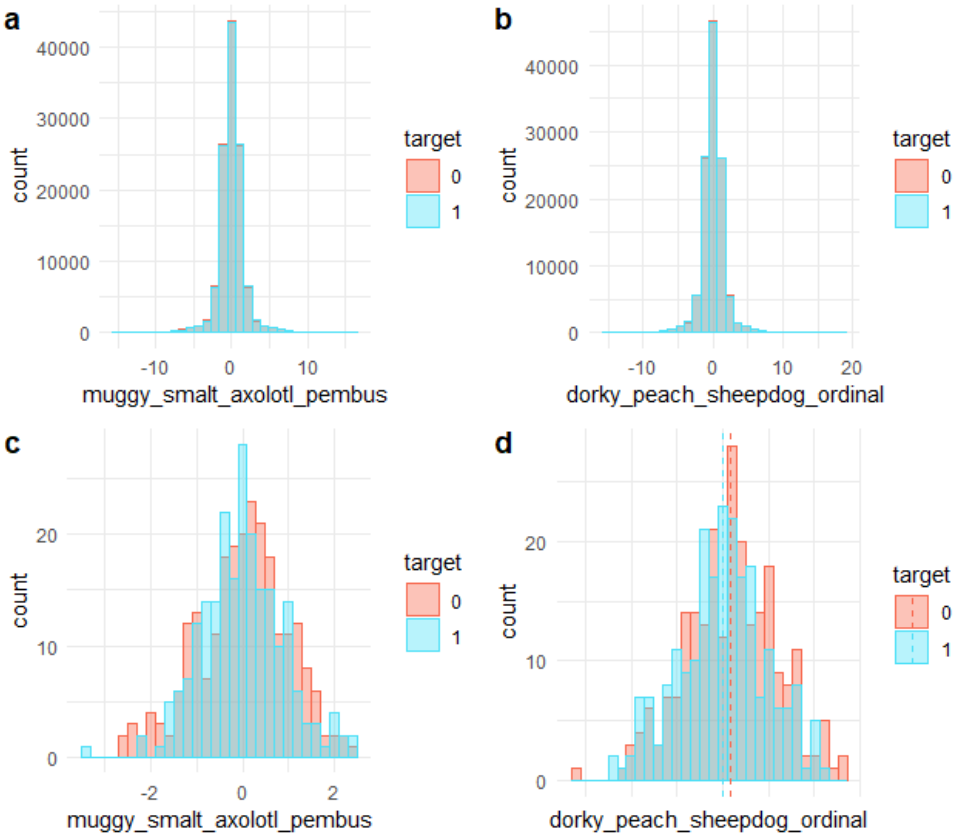


Figure 6: First two variables' distributions before (a and b) and after (c and d)

Figure 7 shows the target=1 proportion histogram provides a strong evidence that the train dataset is in fact aggregations of a number of smaller datasets; each smaller dataset has a different proportion of classes in the dependent variable.

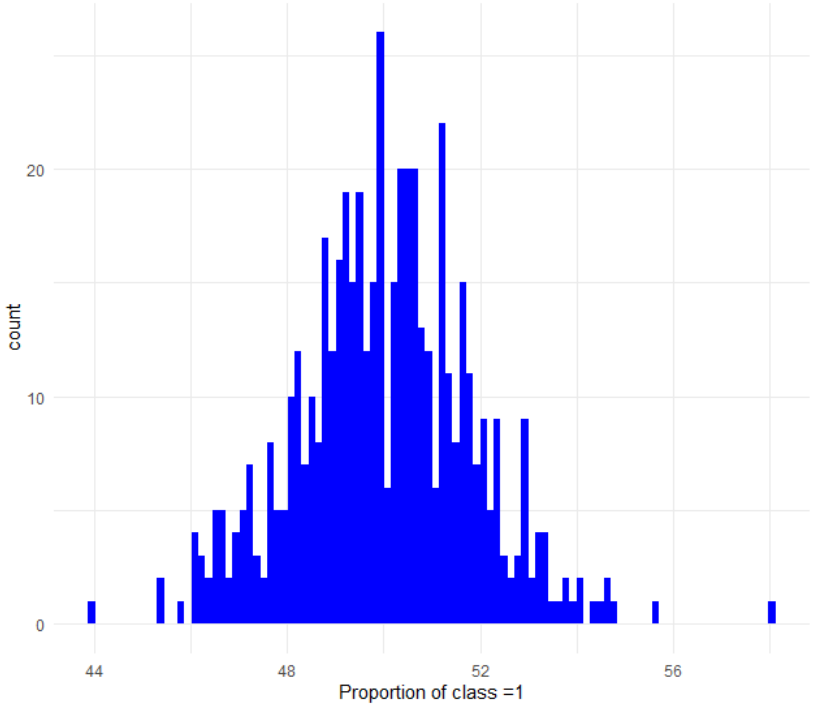


Figure 7: Proportion of target (class = 1)

3.3 Analytical Methods

3.3.1 Methods

I developed two sets of predictive models. The first model is to predict the probabilities of target status in the problem. The second model is to predict the probabilities of target status. Feature selection method such as univariate analysis with p-value<0.1 on target, PCA with best number of components and so on were performed. An interesting phenomenon is that variable standard deviation is either around 1 or 3, and there is no variable has standard deviation between 1.1 and 3.3. Each subset has about 34 predictors selected. Finally, features selection was done by using removing variables with standard deviation under 1.5. This procedure was done through iteration on 512 subsets in R code. The data partition is 70% train and 30% validation.

I used three modeling approaches. I started with a traditional method - Logistic Regression, followed by logistic regression with penalization – Lasso. Since each variable is normal distributed

with different standard deviation, quadratic discriminant analysis (QDA) was performed. It turned out that QDA performed very well. And then I tried KNN and XGBoost to see if we can improve the modeling performance. However, QDA outperformed all the mentioned models. It makes sense that QDA performed the best because QDA is based on the assumption of normal distribution with different standard deviations for each class. The results are shown in Table 2. The modeling accuracy and AUC of ROC curve were used for modeling performance evaluation. The three predictions of test, the probabilities of target status were obtained for test dataset by predicting 512 subsets and aggregating them as well.

3.3.2 Calculation of adjusted posterior probability

If based on the unknown population proportion of "TARGET=1" is only 5%, oversampling should be done on the target variable. Thus, we need to perform posterior probability adjustment here by using below formulas. Results of adjusted probabilities are in the finale column of the attached csv file "EXAM_Testoutput - Yuan Du".

P1 is probability (target =1); P0 is probability (target =0).

A=P1/(50%/5%); B =P0/(50%/95%)

Adjust Probability (target=1) = A/(A+B)

3.4 Results

First Model: The model is to predict the probabilities of target. I found the predictors' standard deviation over 1.5 were the best case. Table 2 shows predictive model performances. All the models perform excellent. The logistic Regression has average 80% AUC in validation. Lasso performed not as good as Logistic Regression with the average AUC of 70%. QDA improved the model performance with average AUC of 96%. QDA outperformed other models on every measure, see Table 2 and appendix Figure 12. The following tables and charts show an example for one subset（subset =279）.

Table 3 shows the statistics of AUC of ROC curve for each model method taken one model as an example. Logistic Regression model has AUC of 0.839, Lasso has AUC of 0.798, and QDA has AUC 0.983. QDA is also the best one in AUC. It can be sure that QDA model is suitable for target prediction. The ROC curve for both validation data are shown in Figure 9.

Table 2: Predictive model performance

	Logistic Regression		**Lasso**		**QDA**	
	Train	Validation	Train	Validation	Train	Validation
Probability Cutoff	0.4959	0.3890	0.5384	0.4744	0.8401	0.6800
Accuracy	81.73%	80.30%	75.96%	77.27%	97.44%	92.42%
Misclassification	18.27%	19.70%	24.04%	22.73%	2.56%	7.58%
Sensitivity	80.79%	85.94%	60.26%	82.81%	96.03%	92.19%
Specificity	82.61%	75.00%	90.68%	72.06%	98.76%	92.65%
Precision:	81.33%	76.39%	85.85%	73.61%	98.64%	92.19%
Prevalence	48.40%	48.48%	48.40%	48.48%	48.40%	48.48%

Table 3: Statistical result summary – AUC of ROC Curves

Predictive model methods	AUC	
	Train	Validation
Logistic Regression	0.897	0.839
Lasso	0.828	0.798
QDA	0.997	0.983

Second model: The model is to predict the probabilities of target status. We can see that the probability of target =1 distribution peaks on both ends. Based on the prediction with different cut-off points, the number of target (1 vs 0) is shown in Table 4.

Table 4: Test data count with different probability cut off

Probability cut off	**N (Target =0)**	**N (Target=1)**
0.5	13100	13114
0.75	13653	12561
0.95	14513	11701

3.4.1 Important variables

An example of variables importance (subset =279) is shown in Figure 8. As we can see, the most important ten factors are cheeky_plum_fox_noise,hasty_ivory_dragonfly_goose, gloppy_cerise_snail_contributor,flimsy_turquoise_fox_kernel,dorky_purple_kiwi_hint,wheezy_red_iguana_entropy,messy_mauve_wolverine_ordinal,lousy_smalt_pinscher_dummy,muggy_pumpkin_scorpion_grandmast, gloppy_turquoise_quoll_goose.

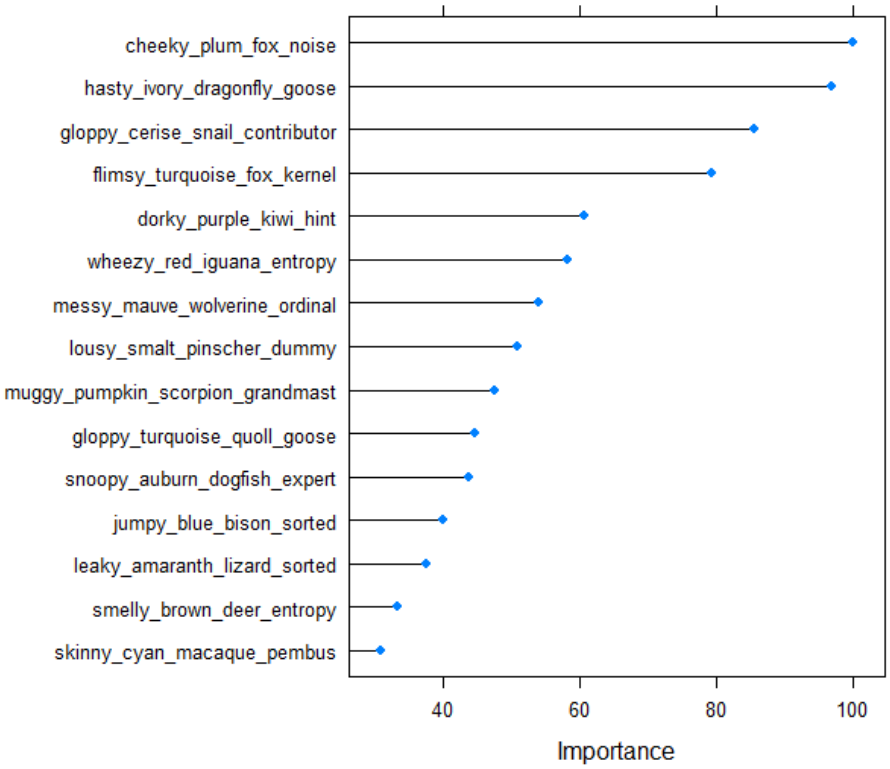
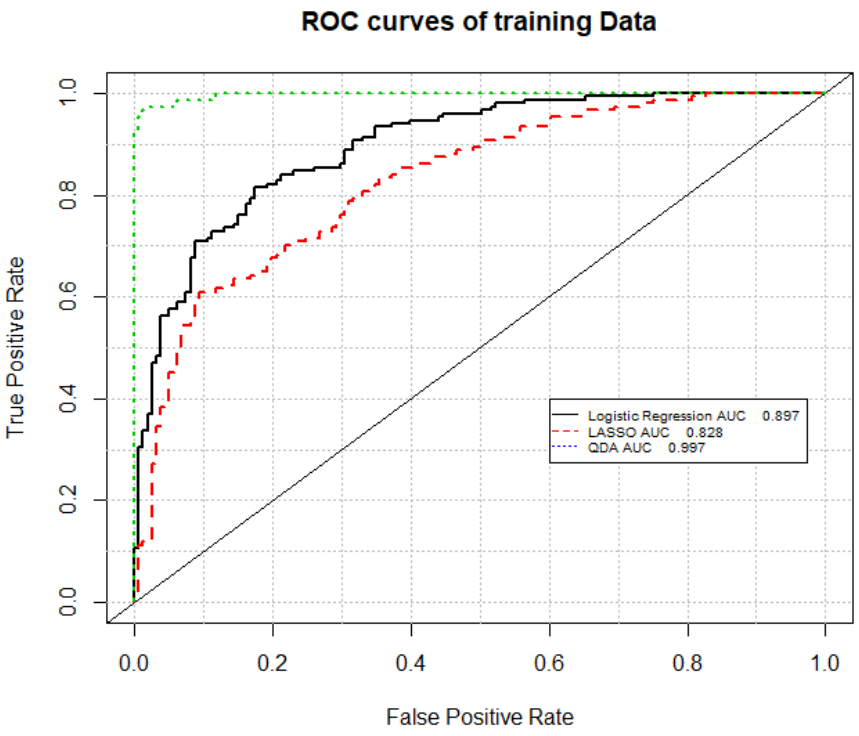


Figure 8: Variable importance

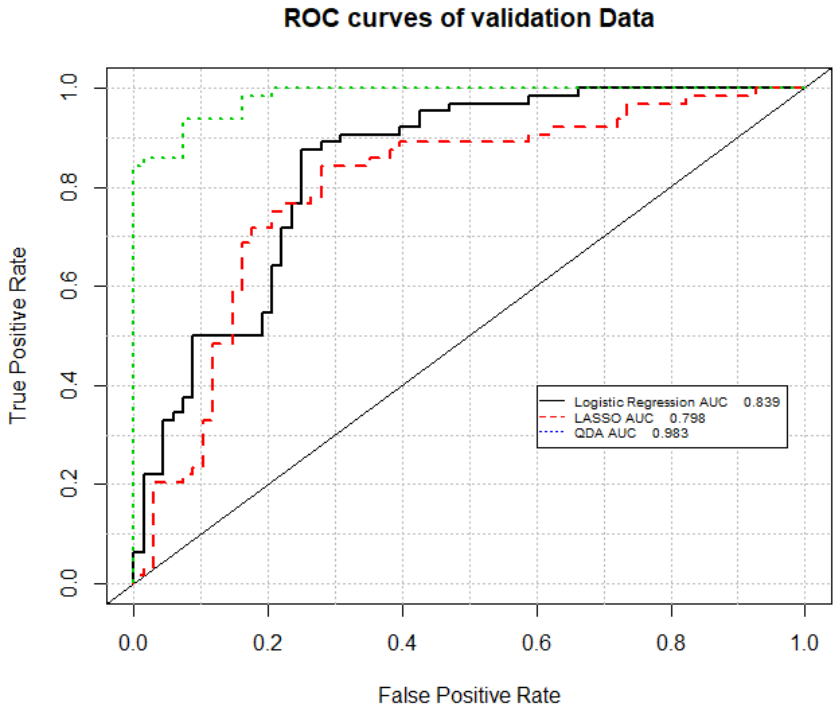
3.4.2 The cut-off probability

Cut-off probability is shown in Table 2, which is the optimal cutoff that balances sensitivity and specificity, for example last model 512 (subset = 279) and QDA turns out has two peaks on both ends.

3.4.3 ROC Curve based on the training data set and validation data set



(a)



(b)

Figure 9: ROC curves on training dataset: partition train (a) and validation (b)

Section 4: Conclusions/Discussion

The conclusions are summarized as below:

(1) I explored the data sets with data visualizations, identified the target variable, integer variable and distributions of numerical variables

(2) I developed two models, which can predict the probabilities of target status. And an idea of target counts by different probability cut-offs.

(3) All models have acceptable prediction accuracy and QDA has an excellent predication accuracy with AUC of 96%.

(4) Further examinations on the unbalanced test probability distribution will be helpful to improve the model predication power.

(5) Potential method such as pseudo labeling can help improve the model performance where the probability is at both ends, such as over 0.99 and under 0.01. Also, Gaussian Mixture Model could also be considered.

Section 5: Appendix

Appendix table of required statistics for each numerical predictor shown in a separate csv file "Appendix Table-Yuan Du.csv". Additional figures (10-13) and R code are in this section:

Figure 10 shows a screen shot of the required statistics for each numerical predictor. Standard deviations are under 2 with high kurtosis.

Vars	Mean	Std.Dev	Min	Q1	Median	Q3	Max	IQR	Skewness	Kurtosis	Out3sd%
baggy_champagne_capuchin_discard	0.02	1.7	-16.97	-0.76	0.01	0.78	16.94	1.53	0.14	8.13	2.45
baggy_copper_oriole_dummy	0	1.65	-16.66	-0.76	0	0.76	17.05	1.52	0.01	8.42	2.4
baggy_coral_bandicoot_unsorted	0	1.85	-17.7	-0.8	0	0.79	15.83	1.59	0	7.12	2.59
baggy_mustard_collie_hint	0	1.8	-18.7	-0.78	0	0.78	18.59	1.56	-0.03	7.83	2.59
beady_aquamarine_affenpinscher_g	0.01	1.82	-19.49	-0.79	0	0.79	18.68	1.57	0.07	7.39	2.55
beady_asparagus_opossum_expert	0.01	1.71	-15.98	-0.77	0	0.77	15.28	1.54	0	7.97	2.5
beady_champagne_bullfrog_grandma	0.01	1.78	-16.14	-0.78	0	0.79	16.47	1.57	0.12	7.51	2.61
beady_lilac_hornet_expert	0	1.78	-15.12	-0.78	0.01	0.78	16.24	1.56	-0.02	7.52	2.53
beady_mauve_frog_distraction	0	1.77	-15.7	-0.78	0	0.77	16.18	1.55	-0.02	7.53	2.56
beady_orange_binturong_golden	0	1.73	-15.8	-0.77	0	0.77	15.16	1.54	-0.03	8.01	2.59

Figure 10: Required statistics

Figure 11 shows a screen shot of the statistics for each numerical predictor after data division/binning within a subset. Standard deviations are around 1 or 3 with low kurtosis.

	Mean	Std.Dev	Min	Q1	Median	Q3	Max	IQR	Skewness	Kurtosis
baggy_champagne_capuchin_discard	0.033467	1.033544	-3.03148	-0.66127	0.076263	0.730511	3.225415	1.390097	-0.06257	-0.06626
baggy_copper_oriole_dummy	-0.04098	1.039836	-2.90405	-0.76606	-0.05484	0.690097	2.656816	1.451103	-0.02827	-0.30947
baggy_coral_bandicoot_unsorted	0.218718	3.714972	-10.7007	-2.24532	0.217104	2.682646	11.13916	4.922405	-0.01589	0.005199
baggy_mustard_collie_hint	0.008289	1.038002	-2.95542	-0.69504	-0.025	0.649651	3.179433	1.343444	0.211697	-0.13777
beady_aquamarine_affenpinscher_g	0.032537	0.972309	-2.57869	-0.62162	0.032722	0.659498	2.847485	1.272518	0.02447	-0.10743
beady_asparagus_opossum_expert	-0.00146	0.979489	-3.10373	-0.62064	0.001074	0.643027	2.943016	1.260934	-0.00538	0.186209
beady_champagne_bullfrog_grandma	0.004618	1.043216	-3.3994	-0.67033	0.022488	0.678484	2.954919	1.346014	-0.12828	-0.0299
beady_lilac_hornet_expert	-0.02135	1.013757	-2.67008	-0.73144	-0.00477	0.680451	3.132474	1.409822	0.076357	-0.23735
beady_mauve_frog_distraction	0.019	3.957332	-10.0474	-2.54054	-0.08054	2.625387	14.98447	5.155284	0.183258	0.219209

Figure 11: Statistics after data division/binning within a subset

Figure 12 shows a random subsets of model performance AUC on validation datasets. QDA has the best AUC in all the subset. And an average AUC of 96%.

	subset	Log	Lasso	QDA	best.model
1	411	0.7402	0.6133	0.9279	QDA
2	235	0.7678	0.7428	0.9526	QDA
3	86	0.8247	0.7286	0.9777	QDA
4	21	0.8812	0.8588	0.9357	QDA
5	215	0.8453	0.7973	0.9617	QDA
6	320	0.7467	0.5	0.9438	QDA
7	272	0.7512	0.7179	0.936	QDA
8	163	0.7838	0.7788	0.9627	QDA
9	169	0.8243	0.6846	0.9376	QDA
10	197	0.7435	0.7587	0.9097	QDA

Figure 12: A Random sets of model performance AUC

Figure 13 shows unusual probability histogram on test data.

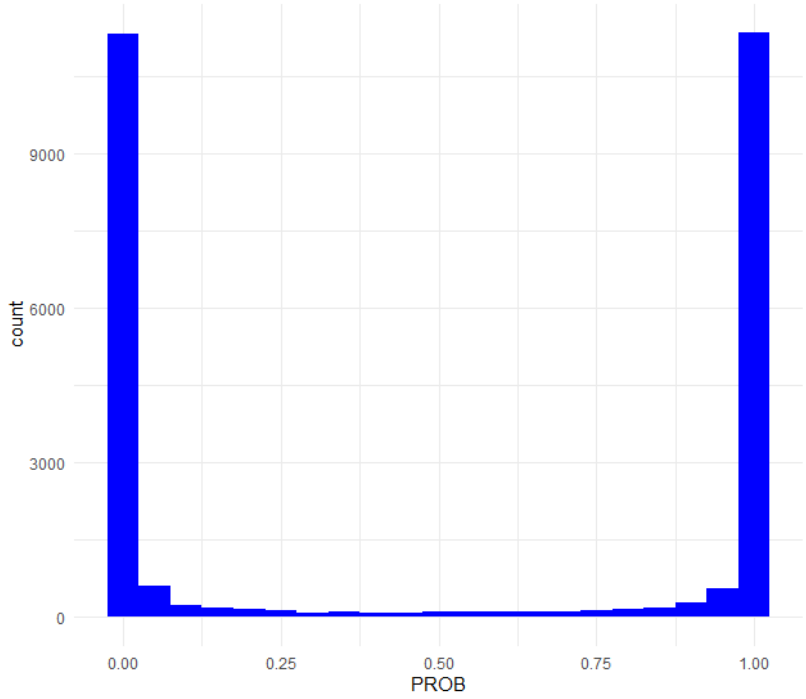


Figure 13: Test data probability histogram

R code:

Instruction to run the code:

1. Please change the directory to the local path that includes the dataset.

For example, if the local directory “Mydata” includes both sub-directories of Training Datasets and Test Datasets. The r code setwd("C:/Datasets/") changes to setwd("C:/Mydata/")

2. For the libraries, it can install automatically if no these libraries. But sometimes you need to select which mirror for downloading.
3. The running time is different dependent on the system. The codes was tested by using R 3.6.2 console within two hours. You can change the “subsets” to a random range to reduce running time, otherwise it is running all 512 models.

```
#####################################################
###########    Qualifying Project 2020   ##########
#####################################################

#################################################
#####         loading  package          #####
#####          if necessary             #####
#####   install.packages automatically   #####
#################################################

if (!require("pacman")) install.packages("pacman")
pacman::p_load(dplyr,tidyverse,ggplot2,verification,
               rstatix, summarytools, caret, MASS, glmnet,
               ROCR, pROC, scales, Matrix)

##### set local path of data source, please modify to local path #####

setwd("C:/Work/Qualify Exam/Project/RawData")

#################################################
#####          Preparing data           #####
#####         EXAM_TRAINData.csv        #####
#####         EXAM_TESTData.csv         #####
#################################################

# train data #
```

```
traindata = read.csv(file = 'EXAM_TRAINData.csv')

# test data #

testdata = read.csv(file = 'EXAM_TESTData.csv')



############################################################
#####              Predictive Modeling            #####
############################################################

#####    Data Partition: 70% training, 30% validation   #####

#setup for models
traindata$target = factor(traindata$target, levels = c(0, 1))
subsets = unique(traindata$wheezy_copper_turtle_magic[])
set.seed(2020)
subsets = sample(0:(length(subsets)-1), 512)

# Matrix to store the performace of the different learning methods for each subset
model.perf = matrix(rep(0, 4*length(subsets)), ncol = 4)#subset, 3 models
colnames(model.perf) = c("subset", "Log", "Lasso","QDA")
model.perf[, "subset"] = subsets #define column subset 0:511

#Build 512 seperate models; In order to save running time, you can sample a few models to see
the performance. Here I only choose the last model.
system.time(for (i in subsets) {
```

```
train2 = traindata[traindata$wheezy_copper_turtle_magic==i,] %>% as.data.frame(.)



# Feature selection (Use about 34 of 255 features, based on variance>1.5)
stattrain2 <- data.frame(summarytools::descr(train2[,!(names(train2) %in%
c("EXID","target"))], transpose = TRUE))
stattrain2 <-  cbind(stattrain2,Vars = rownames(stattrain2))
train2names<-stattrain2 %>% filter(Std.Dev>1.5) %>% dplyr::select(Vars)
train2names<- levels(droplevels(train2names$Vars))

intrain<- createDataPartition(train2$target,p=0.7,list=FALSE)
set.seed(2020)
training<- train2[intrain,]
validating<- train2[-intrain,]

tr=training[names(train2) %in% c(train2names,"target")]
vd=validating[names(train2) %in% c(train2names,"target")]

#for variable importance plot
dtrain<- tr  %>%
  mutate(target = factor(target,
                         labels = make.names(levels(target))))

#Logistic Regression
logModel = glm(target ~ .,family=binomial(link="logit"),data=tr)
prob.log = predict(logModel,newdata=vd[,-(length(train2names)+1)],type='response')
pred.log = prediction(as.numeric(prob.log), as.numeric(vd$target))
perf.log = performance(pred.log, "auc")
```

```
  perf.log = perf.log@y.values %>% unlist() %>% round(., 4)

  ## LASSO ##
  x<-model.matrix(target~.,tr)[,-1]
  y<-tr$target
  xvd<- model.matrix(target~.,vd)[,-1]
  set.seed(2020)
  LaSSOcv = cv.glmnet(x, y,alpha=0, family = "binomial", nfolds = 5)
  best.lambda = LaSSOcv$lambda.min
  LaSSOModel = glmnet(x, y, family = "binomial", lambda = best.lambda)
  prob.lasso<- predict(LaSSOModel,newx = xvd, type = "response")
  pred.lasso = prediction(as.numeric(prob.lasso), as.numeric(vd$target))
  perf.lasso = performance(pred.lasso, "auc")
  perf.lasso = perf.lasso@y.values %>% unlist() %>% round(., 4)

  #QDA
  qdaModel = qda(target ~ .,data=tr)
  prob.qda = predict(qdaModel,vd)$posterior[,2]
  pred.qda = prediction(as.numeric(prob.qda), as.numeric(vd$target))
  perf.qda = performance(pred.qda, "auc")
  perf.qda = perf.qda@y.values %>% unlist() %>% round(., 4)

  model.perf[model.perf[, "subset"] == i, "Log"] = perf.log
  model.perf[model.perf[, "subset"] == i, "Lasso"] = perf.lasso
  model.perf[model.perf[, "subset"] == i, "QDA"] = perf.qda
}
)
```

```
#Model performance: QDA outformed all models in all subsets
model.perf = cbind(model.perf, best.model = apply(model.perf[, 2:4], 1,
function(x){which.max(x) %>% names()}))
model.perf[, "best.model"]

## Model accuracy example last Model 512 ##
#log training
logTrPred=predict(logModel,newdata=tr[,-(length(train2names)+1)],type='response')
pred.logtr=prediction(as.numeric(logTrPred), as.numeric(tr$target))
acc.logperf = performance(pred.logtr, measure = "acc")
ind.log = which.max( slot(acc.logperf, "y.values")[[1]] )
cutoff.log = slot(acc.logperf, "x.values")[[1]][ind.log]
prob.logtr <- ifelse(logTrPred >cutoff.log,1,0)
caret::confusionMatrix(factor(prob.logtr,levels = c(0, 1)), tr$target,positive="1")

#log validation

acc.logperf = performance(pred.log, measure = "acc")
ind.log = which.max( slot(acc.logperf, "y.values")[[1]] )
cutoff.log = slot(acc.logperf, "x.values")[[1]][ind.log]
prob.logvd <- ifelse(prob.log >cutoff.log,1,0)
caret::confusionMatrix(factor(prob.logvd,levels = c(0, 1)), vd$target,positive="1")

#lasso training
lassoTrPred=predict(LaSSOModel,newx = x, type = "response")
pred.lassotr=prediction(as.numeric(lassoTrPred), as.numeric(tr$target))
acc.lassoperf = performance(pred.lassotr, measure = "acc")
ind.lasso = which.max( slot(acc.lassoperf, "y.values")[[1]] )
```

```
cutoff.lasso = slot(acc.lassoperf, "x.values")[[1]][ind.lasso]
prob.lassotr <- ifelse(lassoTrPred >cutoff.lasso,1,0)
caret::confusionMatrix(factor(prob.lassotr,levels = c(0, 1)), tr$target,positive="1")

#lasso validation

acc.lassoperf = performance(pred.lasso, measure = "acc")
ind.lasso = which.max( slot(acc.lassoperf, "y.values")[[1]] )
cutoff.lasso = slot(acc.lassoperf, "x.values")[[1]][ind.lasso]
prob.lassovd <- ifelse(prob.lasso >cutoff.lasso,1,0)
caret::confusionMatrix(factor(prob.lassovd,levels = c(0, 1)), vd$target,positive="1")



#QDA training
qdaTrPred = predict(qdaModel,tr)$posterior[,2]
pred.qdatr=prediction(as.numeric(qdaTrPred), as.numeric(tr$target))
acc.qdaperf = performance(pred.qdatr, measure = "acc")
ind.qda = which.max( slot(acc.qdaperf, "y.values")[[1]] )
cutoff.qda = slot(acc.qdaperf, "x.values")[[1]][ind.qda]
prob.qdatr <- ifelse(qdaTrPred >cutoff.qda,1,0)
caret::confusionMatrix(factor(prob.qdatr,levels = c(0, 1)), tr$target,positive="1")

#QDA validation

acc.qdaperf = performance(pred.qda, measure = "acc")
ind.qda = which.max( slot(acc.qdaperf, "y.values")[[1]] )
cutoff.qda = slot(acc.qdaperf, "x.values")[[1]][ind.qda]
prob.qdavd <- ifelse(prob.qda >cutoff.qda,1,0)
```

```
caret::confusionMatrix(factor(prob.qdavd,levels = c(0, 1)), vd$target,positive="1")

#ROC train last model512
roc.plot(x = tr$target == "1", pred = cbind(logTrPred,lassoTrPred,qdaTrPred),
        show.thres = FALSE, legend = TRUE, main="ROC curves of training Data",
        xlab = "False Positive Rate", ylab="True Positive Rate",
        leg.text = c("Logistic Regression AUC","LASSO AUC","QDA AUC"))$roc.vol

#ROC validation last model512
roc.plot(x = vd$target == "1", pred = cbind(prob.log,prob.lasso,prob.qda),
        show.thres = FALSE, legend = TRUE, main="ROC curves of validation Data",
        xlab = "False Positive Rate", ylab="True Positive Rate",
        leg.text = c("Logistic Regression AUC","LASSO AUC","QDA AUC"))$roc.vol

## variable importance ##
Ctrl <- trainControl(summaryFunction=twoClassSummary,
                     classProbs=TRUE)
appr.qda <- train(target~., data=dtrain, method="qda",
                  trControl=Ctrl, preProc = c("center","scale"),
                  metric="ROC")
varim<-varImp(appr.qda)
plot(varim, top=15)

### predict test data ###
#setup test output for model output
train = subset(traindata, select = -EXID)
test.id.val = testdata$EXID
key=testdata$wheezy_copper_turtle_magic
```

```
test = subset(testdata, select=-EXID)
target = rep(0, nrow(testdata))
out = data.frame(test.id.val, key, target)

#run 512 models for testdata
system.time(for (i in subsets) {

  train2 = traindata[traindata$wheezy_copper_turtle_magic==i,] %>% as.data.frame(.)
  test2 = testdata[testdata$wheezy_copper_turtle_magic == i, ] %>% as.data.frame(.) %>%
    dplyr::select(-wheezy_copper_turtle_magic)



  # Feature selection (Use 34 of 255 features, based on variance>1.5)
  stattrain2 <- data.frame(summarytools::descr(train2[,!(names(train2) %in%
c("EXID","target"))], transpose = TRUE))
  stattrain2 <-  cbind(stattrain2,Vars = rownames(stattrain2))
  train2names<-stattrain2 %>% filter(Std.Dev>1.5) %>% dplyr::select(Vars)
  train2names<- levels(droplevels(train2names$Vars))


  stattest2 <- data.frame(summarytools::descr(test2[,!(names(test2) %in% "EXID")], transpose =
TRUE))
  stattest2 <-  cbind(stattest2,Vars = rownames(stattest2))
  test2names<-stattest2 %>% filter(Std.Dev>1.5) %>% dplyr::select(Vars)
  test2names<- levels(droplevels(test2names$Vars))


  intrain<- createDataPartition(train2$target,p=0.7,list=FALSE)
  set.seed(2020)
  training<- train2[intrain,]
```

```
  validating<- train2[-intrain,]

  tr=training[names(train2) %in% c(train2names,"target")]
  vd=validating[names(train2) %in% c(train2names,"target")]
  test3=test2[names(test2) %in% test2names]

  #QDA
  qdaModel = qda(target ~ .,data=tr)
  prob.qda = predict(qdaModel,vd)$posterior[,2]
  pred.qda = prediction(as.numeric(prob.qda), as.numeric(vd$target))
  perf.qda = performance(pred.qda, "auc")
  perf.qda = perf.qda@y.values %>% unlist() %>% round(., 4)


  prob = predict(qdaModel, newdata = test3)$posterior[, "1"]
  out$target[out$key == i] = prob

}
)

#############################################################################
#####    Apply Predictive Modeling in test data-write all results     #####
#############################################################################

# Finalizing the output
out = subset(out, select = -key)
output=out %>% rename(EXID=test.id.val, PROB=target)%>%
mutate(P_Target=1,ADJ_PROB=PROB)%>%dplyr::select(EXID,P_Target,PROB,ADJ_PROB)
```

```
write.csv(output, file = "EXAM_Testoutput.csv")

# Find out cutoff
out.status  <- ifelse(out$target > 0.5,1,0)
table(out.status)
out.status  <- ifelse(out$target > 0.75,1,0)
table(out.status)
out.status  <- ifelse(out$target > 0.95,1,0)
table(out.status)
```