**Data Mining Code  
Case study: Home credit default risk**

Anurag Mishra 17125007  
Archeesh Kalra 17125009  
Bahul Dandona 17125010  
Danish Nawaz 18114008  
Sandipan Bhattacharjee 17125050  
Yatin Kundra 17125069

**# Libraries**

load.libraries <- c('plyr', 'dplyr','data.table', 'readxl', 'reshape2', 'stringr', 'stringi', 'ggplot2', 'tidyverse','gridExtra','matrixStats','lubridate','corrplot','e1071','xgboost','caret','zoo','factoextra','plotly','DT')

install.lib <- load.libraries[!load.libraries %in% installed.packages()]

for(libs in install.lib) install.packages(libs, dependences = TRUE)  
sapply(load.libraries, require, character = TRUE)

**## Cleaning the data**

dt1$DAYS\_EMPLOYED <- replace(dt1$DAYS\_EMPLOYED,dt1$DAYS\_EMPLOYED == 365243,NA)

#### Cleaning Rule 2

```{r include=FALSE}

col\_neg <- unlist(dt1[, sapply(dt1, FUN = function(x) all(x <= 0, na.rm = TRUE))])

dt1\_abs <- setDT(dt1)[,..col\_neg]

dt1\_abs <- abs(dt1\_abs)

rm\_col <- colnames(dt1\_abs)

dt1 <- as.data.frame(dt1)[, !(colnames(dt1) %in% rm\_col)]

dt1 <- cbind(dt1\_abs,dt1)

**## Data Pre Processing**

**## Data Transformations**

**#### Skewness of the data**

numeric\_list <- unlist(lapply(dt1, is.numeric))

dt1\_num <- setDT(dt1)[,..numeric\_list]

**##### Exploratory (Exhaustive) Data Analysis {.tabset .tabset-fade .tabset-pills}**

###### Plots1

```{r fig1, fig.height = 20, fig.width = 10}

doPlots(dt1\_num, plotHist, ii = 1:20)

```

###### Plots2

```{r fig2, fig.height = 20, fig.width = 10}

doPlots(dt1\_num, plotHist, ii = 21:41)

```

###### Plots3

```{r fig3, fig.height = 20, fig.width = 10}

#doPlots(dt1\_num, plotHist, ii = 42:62)

```

###### Plots4

```{r fig4, fig.height = 20, fig.width = 10}

#doPlots(dt1\_num, plotHist, ii = 63:83)

```

###### Plots5

```{r fig5, fig.height = 20, fig.width = 10}

#doPlots(dt1\_num, plotHist, ii = 84:106)

```

**#### Skewness of the data (cont'd)**

\*Lets look at the skewness of all the columns. The table below shows a summary\*

```{r}

skewValues <- as.data.frame(apply(dt1\_num, 2, function(x) skewness(x, na.rm = TRUE)))

colnames(skewValues)[1] <- "skew\_values"

skewValues <- index\_to\_col(skewValues,'Column')

skewValues <- setDT(skewValues)[order (skew\_values, decreasing = TRUE)]

skewValues[sample(1:nrow(skewValues), size = nrow(skewValues)),] %>%

datatable(filter = 'top', options = list(

pageLength = 15, autoWidth = F

))

```

```{r}

BoxCoxValues <- apply(dt1\_num, 2, function(x) BoxCoxTrans(x, na.rm = TRUE))

x = list()

for (i in 1:ncol(dt1\_num)){

lambda <- BoxCoxValues[[i]][[1]]

x[[i]] <- lambda

}

lambda = do.call(rbind, x)

lambda\_df <- as.data.frame(cbind(colnames(dt1\_num),lambda))

colnames(lambda\_df)[1] <- "Column"

colnames(lambda\_df)[2] <- "lambda"

knitr::kable(setDT(lambda\_df)[!is.na(lambda)])

```

```{r}

preProcValues <- preProcess(dt1, method = "BoxCox")

preProcValues

dt1\_tran <- predict(preProcValues, dt1)

#Recreate numeric list with new dt1\_tran

numeric\_list <- unlist(lapply(dt1\_tran, is.numeric))

dt1\_num <- setDT(dt1\_tran)[,..numeric\_list]

```

```{r}

col\_trans <- lambda\_df[!is.na(lambda)]$Column

i = 5

x <- list(

title = as.character(col\_trans[i])

)

p1 <- plot\_ly(x = ~setDT(dt1)[,get(as.character(col\_trans[i]))], type = "histogram", autobinx = FALSE) %>% layout(showlegend = FALSE)

p2 <- plot\_ly(x = ~setDT(dt1\_tran)[,get(as.character(col\_trans[i]))], type = "histogram", autobinx = FALSE) %>% layout(showlegend = FALSE)

subplot(p1,p2)

```

**##### Transformed Predictors (Variables) {.tabset .tabset-fade .tabset-pills}**

**###### Without Transformation**

```{r fig t1, fig.height = 20, fig.width = 10}

doPlots(as.data.frame(dt1)[, (colnames(dt1) %in% as.character(col\_trans))], plotHist, ii = 1:length(col\_trans))

```

###### After Transformation

```{r fig t2, fig.height = 20, fig.width = 10}

doPlots(as.data.frame(dt1\_tran)[, (colnames(dt1\_tran) %in% as.character(col\_trans))], plotHist, ii = 1:length(col\_trans))

```

#### Transformation to resolve Outliers

### Handling Missing Values

\*Finding the % of missing values for all columns\*

```{r figmv, fig.height = 7, fig.width = 10}

mv <- as.data.frame(apply(dt1\_tran, 2, function(col)sum(is.na(col))/length(col)))

colnames(mv)[1] <- "missing\_values"

mv <- index\_to\_col(mv,'Column')

mv <- setDT(mv)[order (missing\_values, decreasing = TRUE)]

ggplot (mv[1:40,], aes (reorder(Column, missing\_values), missing\_values)) + geom\_bar (position = position\_dodge(), stat = "identity") + coord\_flip () + xlab('Columns') + ylab('Missing Value %')

```

```{r }

dt1\_num2 <- na.aggregate(dt1\_num)

```

### Data Reduction and Feature Extraction

```{r}

regexp <- "[[:digit:]]+"

pcaObject <- prcomp(dt1\_num2, scale = TRUE, center = TRUE)

eig\_tb <- cbind(Dimensions = rownames(get\_eig(pcaObject)), get\_eig(pcaObject))

ts <- setDT(eig\_tb)[cumulative.variance.percent > 80][1,1]

ts <- str\_extract(as.character(ts[[1]]), regexp)

n <- as.numeric(ts)

col\_list <- list()

for (i in 1:n){

col\_list[i]<-paste('rotation.PC',i, sep="")

}

pca\_df <- as.data.frame(pcaObject[2])

pca\_df <- pca\_df[,colnames(pca\_df) %in% col\_list]

pca\_df <- cbind(Features = rownames(pca\_df), pca\_df)

pca\_df <- setDT(pca\_df)[order (rotation.PC1, decreasing = TRUE)]

```{r echo=FALSE}

#comp <- data.frame(pcaObject$x[,1:4])

#plot(comp, col=dt1\_tran$TARGET, pch=16)

```

### Removing Predictors

#### Zero Variance Predictors

```{r}

nzv <- nearZeroVar(dt1,saveMetrics= TRUE)

nzv <- index\_to\_col(nzv,"Column")

nzv\_tb <- setDT(nzv)[nzv == TRUE | zeroVar ==TRUE]

nzv\_tb[sample(1:nrow(nzv\_tb), size = nrow(nzv\_tb)),] %>%

datatable(filter = 'top', options = list(

pageLength = 15, autoWidth = T

))

#### Between Predictor-Correlations

```{r echo=FALSE}

dt1\_num\_corr <- dt1\_num

colnames(dt1\_num\_corr)[1:ncol(dt1\_num\_corr)] <- c(1:ncol(dt1\_num\_corr))

correlations <- cor(na.omit(dt1\_num\_corr))

corrplot(correlations, method="square")

```{r}

df\_corr = cor(dt1\_num2, use = "pairwise.complete.obs")

hc = findCorrelation(df\_corr, cutoff=0.80)

hc = sort(hc)

dt1\_num3 = as.data.frame(dt1\_num2)[,-c(hc)]

rm\_col\_hc <- setdiff(colnames(dt1\_num2),colnames(dt1\_num3))

rm\_col\_hc

```

#Highly correlated vairables table format

df\_corr2 <- df\_corr %>%

as.data.frame() %>%

mutate(var1 = rownames(.)) %>%

gather(var2, value, -var1) %>%

arrange(desc(value)) %>%

group\_by(value)

corr\_tb <- setDT(df\_corr2)[abs(value) > 0.8 & var1 != var2 & var1 != "Ttl\_Rating" & var2 != "Ttl\_Rating"]

corr\_tb <- corr\_tb[!duplicated(corr\_tb$value),]

l1 <- corr\_tb$var1

l2 <- corr\_tb$var2

corr\_tb[sample(1:nrow(corr\_tb), size = nrow(corr\_tb)),] %>%

datatable(filter = 'top', options = list(

pageLength = 15, autoWidth = T

))

```

##### Scatter Plots (Highly Correlated Variables) {.tabset .tabset-fade .tabset-pills}

###### Scatter1

```{r fig s1, fig.height = 8, fig.width = 10}

doPlotsCorr(dt1\_num2,plotCorr,l1,l2,1:6)

```

###### Scatter2

```{r fig s2, fig.height = 15, fig.width = 10}

#doPlotsCorr(dt1\_num2,plotCorr,l1,l2,13:27)

```

###### Scatter3

```{r fig s3, fig.height = 15, fig.width = 10}

#doPlotsCorr(dt1\_num2,plotCorr,l1,l2,71:83)

```

###### Scatter4

```{r fig s4, fig.height = 15, fig.width = 10}

#doPlotsCorr(dt1\_num2,plotCorr,l1,l2,58:70)

```

\*Removing all the columns identified as highly correlated and/or nzv\*

```{r}

rm\_col\_all <- append(rm\_col\_hc,rm\_col\_nzv)

dt1\_tran <- as.data.frame(dt1\_tran)[, !(colnames(dt1\_tran) %in% rm\_col\_all)]

```

\*Number of columns remaining are `r dim(dt1)[2] - dim(dt1\_tran)[2]` initially from `r dim(dt1)[2]`\*

```{r echo=FALSE}

#Recreate numeric list with new dt1\_tran

numeric\_list <- unlist(lapply(dt1\_tran, is.numeric))

dt1\_num <- setDT(dt1\_tran)[,..numeric\_list]

```

##### Transformed Predictors (Variables) {.tabset .tabset-fade .tabset-pills}

\*Now that all the pre processing for numeric variables is done, lets have a look at the density plot to compare the target variable\*

###### Density Plot 1

```{r fig d1, fig.height = 20, fig.width = 10}

doPlots(dt1\_num2, plotDen, ii = 1:20)

```

###### Density Plot 2

```{r fig d2, fig.height = 20, fig.width = 10}

doPlots(dt1\_num2, plotDen, ii = 21:40)

```

### Adding Predictors

```{r echo= FALSE}

non\_numeric\_list <- unlist(lapply(dt1\_tran, is.character))

dt1\_non\_num <- setDT(dt1\_tran)[,..non\_numeric\_list]

```

##### BarPlot of the Categorical and Target Variable {.tabset .tabset-fade .tabset-pills}

###### BarPlot 1

```{r fig b1, fig.height = 10, fig.width = 10}

doPlots(dt1\_non\_num, plotBar, ii = 1:9)

```

###### BarPlot 2

```{r fig b2, fig.height = 10, fig.width = 10}

doPlots(dt1\_non\_num, plotBar, ii = c(9,11,13:16))

```

###### BarPlot 3

```{r fig b3, fig.height = 10, fig.width = 10}

grid.arrange(plotBar(dt1\_non\_num, 10),plotBar(dt1\_non\_num, 12), ncol=1, nrow=2)

```

```{r}

dt1\_non\_num <- cbind(dt1\_non\_num,dt1\_tran[,'TARGET'])

dummies <- dummyVars(TARGET ~ ., data = dt1\_non\_num, drop2nd = TRUE)

dt1\_non\_num\_dum <- predict(dummies, newdata = dt1\_non\_num)

```

### Binning Predictors

```{r}

dt1\_preproc <- cbind(dt1\_non\_num\_dum,dt1\_num)

mv <- as.data.frame(apply(dt1\_preproc, 2, function(col)sum(is.na(col))/length(col)))

colnames(mv)[1] <- "missing\_values"

mv <- index\_to\_col(mv,'Column')

mv <- setDT(mv)[order (missing\_values, decreasing = TRUE)]

ggplot (mv[1:40,], aes (reorder(Column, missing\_values), missing\_values)) + geom\_bar (position = position\_dodge(), stat = "identity") + coord\_flip () + xlab('Columns') + ylab('Missing Value %')

dt1\_preproc <- na.aggregate(dt1\_preproc)

```

## Over-Fitting and Model Tuning

### Overfitting and its Problems

### Model Tuning

### Data Splitting

\*Going forward I will be using a small sample since the dataset is too big for processing\*

```{r include=FALSE}

set.seed(1234)

dt1\_preproc\_sample <- setDT(dt1\_preproc)[sample(nrow(dt1\_preproc), round(nrow(dt1\_preproc)\*0.01,0)),]

```

```{r, echo=T}

# control <- rfeControl(functions=rfFuncs, method="cv", number=3)

# trainctrl <- trainControl(classProbs= TRUE, summaryFunction = twoClassSummary)

#

# results <- rfe(as.data.frame(dt1\_preproc\_sample)[,-c(153)],

# as.data.frame(dt1\_preproc\_sample)[,c(153)], sizes=c(1:100),

# rfeControl=control,

# method="rf",

# metric = "AUC",

# trControl = trainctrl)

# print(results)

# predictors(results)

# plot(results, type=c("g", "o"))

```{r Boruta}

#boruta.train <- Boruta(TARGET~., data = dt1\_preproc, doTrace = 2)

#print(boruta.train)

```

#cols\_to\_keep <- c(predictors(results),"TARGET")

cols\_to\_keep <- c('FLAG\_OWN\_CARN','`ORGANIZATION\_TYPEIndustry: type 1`','DAYS\_ID\_PUBLISH','SK\_ID\_CURR','REG\_CITY\_NOT\_LIVE\_CITY','YEARS\_BEGINEXPLUATATION\_MODE','COMMONAREA\_MODE','FLOORSMAX\_MODE','LIVINGAPARTMENTS\_MODE','YEARS\_BUILD\_MEDI','CODE\_GENDERM','OCCUPATION\_TYPEWaiters/barmen staff','TARGET','EXT\_SOURCE\_1','EXT\_SOURCE\_2','EXT\_SOURCE\_3')

dt1\_preproc\_sample <- as.data.frame(dt1\_preproc\_sample)[, (colnames(dt1\_preproc\_sample) %in% cols\_to\_keep)]

```{r include=FALSE}

predictors <- setDT(dt1\_preproc\_sample)[,-c('TARGET')]

classes <- as.factor(dt1\_preproc\_sample$TARGET)

trainingRows <- createDataPartition(y=classes, p = 0.80, list =FALSE)

trainPredictors <- predictors[trainingRows,]

trainclasses <- classes[trainingRows]

testPredictors <- predictors[-trainingRows,]

testClasses <- classes[-trainingRows]

```

### Resampling Techniques

#### K-Fold Cross-Validation

```{r}

cvSplits <- createFolds(trainclasses, k = 10, returnTrain = TRUE)

```

#### Repeated Training Test Splits

```{r}

repeatedSplits <- createDataPartition(trainclasses, p =0.8, times = 3)

```

#### The Bootstrap

```{r}

bsSplits <- createResample(trainclasses, times = 10, list = TRUE)

```

### Data Splitting Recommendations

### Choosing Between Models

###Running a Simple model

```{r DataPartition}

dt1\_preproc\_sample <- mutate(dt1\_preproc\_sample, TARGET = ifelse(TARGET == 0,'Yes',"No"))

dt1\_preproc\_sample$TARGET <- as.factor(dt1\_preproc\_sample$TARGET)

inTrain <- createDataPartition(dt1\_preproc\_sample$TARGET, p = .8)[[1]]

dtTrain <- dt1\_preproc\_sample[ inTrain, ]

dtTest <- dt1\_preproc\_sample[-inTrain, ]

```

4

```{r echo=T, results='hide'}

traincntrl <- trainControl(method = 'repeatedcv',

number = 5,

repeats = 2,

classProbs = TRUE,

sampling = "down",

summaryFunction = twoClassSummary)

```

Running a KNN model

```{r}

trainPredictors <- as.matrix(trainPredictors)

knnFit <- train(TARGET ~.,

data = dtTrain,

method = "knn",

preProc = c("center", "scale"),

tuneGrid = data.frame(.k = 3:6),

trControl = traincntrl)

knnFit$results

```

Running a SVM model

```{r SVM}

svmFit <- train(TARGET ~.,

data = dtTrain,

method = 'svmRadial',

preProc = c('center','scale'),

tuneLength = 7,

trControl = traincntrl)

svmFit

plot(svmFit, scales = list(x=list(log =2)))

predictClasses <- predict(svmFit, dtTest)

predictProbs <- predict(svmFit, newdata = dtTest, type = "prob")

```

```{r}

# svmFit$results %>%

# mutate(accuracySD\_low = Accuracy - 2\*(AccuracySD/sqrt(svmFit$control$number \* svmFit$control$repeats)),

# accuracySD\_high = Accuracy + 2\*(AccuracySD/sqrt(svmFit$control$number \* svmFit$control$repeats))) %>%

# ggplot(aes(x = C)) +

# geom\_line(aes(y = Accuracy)) +

# geom\_point(aes(y = Accuracy)) + theme\_classic() +

# scale\_x\_log10() + #correct spacing of the cost parameter

# ylim(0.50, 0.70) + #set correct y-axis

# geom\_errorbar(aes(ymin=accuracySD\_low, ymax=accuracySD\_high),

# colour="gray50",

# width=.1) +

# labs(title="Estimates of prediction accuracy\nwith 2 SD errror bars")

#

```{r LogisticRegression}

logisticReg <- train(TARGET ~.,

data = dtTrain,

method = 'glm',

trControl = traincntrl)

```

```{r Comparingresults}

resamp <- resamples(list(SVM = svmFit, Logistic = logisticReg, KNN = knnFit))

summary(resamp)

```

#Measuring Performance in Regression Models

###Variance-Bias Trade off

#Nonlinear Regression Models

###Support Vector Machines (SVM)

```{r SVM\_Tuning}

svmFitRadial <- svmFit

svmFitLinear <- train(TARGET ~.,

data = dtTrain,

method = 'svmLinear',

preProc = c('center','scale'),

metric = "ROC",

tuneLength = 7,

trControl = traincntrl)

# svmFitPoly <- train(TARGET ~.,

# data = dtTrain,

# method = 'svmPoly',

# preProc = c('center','scale'),

# tuneLength = 7,

# trControl = traincntrl)

```

```{r}

resamp <- resamples(list(SVM\_Radial = svmFit, SVM\_Linear = svmFitLinear))

summary(resamp)

```

###K-Nearest Neighbors (KNN)

```{r KNN\_Tuning}

knnFit <- train(TARGET ~.,

data = dtTrain,

method = "knn",

preProc = c("center", "scale"),

metric = "ROC",

tuneGrid = data.frame(.k = 1:20),

trControl = traincntrl)

knnFit$results

#Measuring Performance in Classification Models

###Class Predictions

```{r}

dtTest$svmFitLinearclass <- predict(svmFitLinear, dtTest)

dtTest$svmFitLinearprobs <- predict(svmFitLinear, newdata = dtTest , type = "prob")

dtTest$logclass <- predict(logisticReg, dtTest)

dtTest$logprobs <- predict(logisticReg, newdata = dtTest , type = "prob")

```

```{r}

calCurve <- calibration(TARGET ~ svmFitLinearprobs[,1] + logprobs[,1], data = dtTest)

calCurve

xyplot(calCurve, auto.key = list(columns = 2))

```

```{r}

confusionMatrix(data = dtTest$svmFitLinearclass,

reference = dtTest$TARGET,

positive = "Yes")

```

```{r}

confusionMatrix(data = dtTest$logclass,

reference = dtTest$TARGET,

positive = "Yes")

```

###Evaluating Class Probabilities

```{r}

library(pROC)

rocCurve <- roc(response = dtTest$TARGET, predictor = dtTest$svmFitLinearprobs[,1], levels = rev(levels(dtTest$TARGET)))

plot(rocCurve, legacy.axes = TRUE)

auc(rocCurve)

```

#Logistic Regression

loan.model <- subset(train\_data\_new, select = c(4,8,9,18,42:44,2))

# No missing values

dim(loan.model)

library(tidyr)

loan.model<-loan.model%>% drop\_na()

anyNA(loan.model)

#partitioning the dataset

#Splitting data set into training and test set

## 75% of the sample size

smp\_size <- floor(0.75 \* nrow(loan.model))

## set the seed to make your partition reproducible

set.seed(123)

train\_ind <- sample(seq\_len(nrow(loan.model)), size = smp\_size)

train <- loan.model[train\_ind, ]

test <- loan.model[-train\_ind, ]

#LOGISTIC REGRESSION

library(glm2)

logistic.regressor <- glm(TARGET ~ ., family = "binomial", data = train)

summary(logistic.regressor)

#Predicting outcomes on test data

prob\_pred <- predict(logistic.regressor, newdata = test, type = "response")

summary(prob\_pred)

#Cut-off value = 0.5

pred\_cut\_off <- ifelse(prob\_pred > 0.5,1,0) #Setting cut-off to be at 0.5

test = data.frame(test,pred\_cut\_off)

table(test$TARGET,pred\_cut\_off )

Accuracy = 100\*sum(diag(table(test$TARGET,pred\_cut\_off))) /nrow(test)

Accuracy

library(prediction)

pred <- prediction(as.numeric(pred\_cut\_off),test$TARGET)

perf <- performance(pred, "tpr", "fpr")

#Printing AUC Value

perf1 <- performance(pred, "auc")

print(perf1@y.values[[1]])

pt = 1 / (1 + exp(-prob\_pred))

pt = 1 - pt

ntest <- nrow(test)

ac\_tot = 100\*(1:ntest) / ntest

pt.ord = order(pt, decreasing=T)

Status\_test = test$TARGET

npos = table(Status\_test)[2]

ac\_pos\_test = 100\*cumsum(Status\_test[pt.ord] == 1) / npos

plot(ac\_tot, ac\_pos\_test, type="l", lwd=2,

main="Concentration Curve")

lines(ac\_tot, ac\_tot, col="gray70", lwd=2)

# ================================================================================

# ROC Curve

# ================================================================================

nneg = ntest - npos

ac\_neg\_test = 100\*cumsum(Status\_test[pt.ord]==0) / nneg

plot(ac\_pos\_test,ac\_neg\_test, type="l", lwd=2, main="ROC Curve", col="blue",xlab="FPR",ylab="TPR")

lines(ac\_neg\_test, ac\_neg\_test, col="grey70", lwd=2)

**##Decision Tree:**

library(rpart)

library(rpart.plot)

fit <- rpart(TARGET~., data = train, method = 'class')

predict\_unseen <-predict(fit,test, type = 'class')

table\_mat <- table(test$TARGET, predict\_unseen)

table\_mat

accuracy\_Test <- sum(diag(table\_mat)) / sum(table\_mat)

print(paste('Accuracy for test', accuracy\_Test))

accuracy\_tune <- function(fit) {

predict\_unseen <- predict(fit,test, type = 'class')

table\_mat <- table(test$TARGET, predict\_unseen)

accuracy\_Test <- sum(diag(table\_mat)) / sum(table\_mat)

accuracy\_Test

logistic.regressor <- glm(TARGET ~ ., family = "binomial", data = train)

prob\_pred <- predict(logistic.regressor, newdata = test, type = "response")

pt = 1 / (1 + exp(-prob\_pred))

pt = 1 - pt

ntest <- nrow(test)

ac\_tot = 100\*(1:ntest) / ntest

pt.ord = order(pt, decreasing=T)

Status\_test = test$TARGET

npos = table(Status\_test)[2]

ac\_pos\_test = 100\*cumsum(Status\_test[pt.ord] == 1) / npos

# ================================================================================

# ROC Curve

# ================================================================================

nneg = ntest - npos

ac\_neg\_test = 100\*cumsum(Status\_test[pt.ord]==0) / nneg

plot(ac\_pos\_test,ac\_neg\_test, type="l", lwd=2, main="ROC Curve", col="blue",xlab="FPR",ylab="TPR")

lines(ac\_neg\_test, ac\_neg\_test, col="grey70", lwd=2)

}

control <- rpart.control(minsplit = 4,

minbucket = round(5 / 3),

maxdepth = 3,

cp = 0)

tune\_fit <- rpart(TARGET~., data =train, method = 'class', control = control)

accuracy\_tune(tune\_fit)