MarketInvoice Task

Sameer Patel

10/12/2020

Introduction

In this exercise, we will build a simple credit model for MarketFinance based on track-record. To do this, we will be using the publicly available Loan book (attached). Let's start with clarifying some of the terms.

We call the company that raises an invoices the seller. The company which the invoice is raised against is called the debtor. In the loan book, you can find the column Seller ID, which contains an id, which can be used to identify different transactions relating to the same seller.

At MarketFinance, our transactions are called trades. Within a trade, the seller receives money (the advance) in return for one or multiple invoices. When a seller submits a trade, they will submit an Expected Payment Date. When a trade is not being paid within the agreed terms, it is said to be delinquent or in arrears. This is the variable that we will be predicting.

Instructions

Task 1

Write a script in R (or Python if you are not familiar with R) to read in the data. Ensure that you will only be working with the Trade Type "Standard".

Notes:

- Create classDate to correclty recognise imported dates
- Substring the Expected.Payment.Date to remove the time & timezone string parts
- Subset data to Trade.Type = "Standard"

```r
# import data
setClass("classDate")
setAs("character", "classDate", function(from) as.Date(from, format = "%Y-%m-%d"))
dt = read.csv("MarketInvoice_Public_Loan_Book.csv", colClasses = c(NA, NA, NA, "classDate",
    NA, "classDate", NA, "classDate", "classDate"))
Expected.Payment.Date = t(as.data.frame(strsplit(dt$Expected.Payment.Date, " ")))
levels(factor(Expected.Payment.Date[, 2]))  #always '00:00:00'
```

```
## [1] "00:00:00"
```

```r
levels(factor(Expected.Payment.Date[, 3]))  #always '+0000'
```

```
## [1] "+0000"
```

```r
dt$Expected.Payment.Date = as.Date(t(as.data.frame(strsplit(dt$Expected.Payment.Date,
    " ")))[, 1], format = "%Y-%m-%d")
```

```r
# subset to trade type Standard
levels(factor(dt$Trade.Type))  #check for typo of Standard
```

```
## [1] "JrlSupplyChainFinance" "LicenceFee"            "MultiDebtor"
## [4] "PurchaseOrder"         "Standard"              "WholeLedger"
```

```r
dt2 = dt[which(dt$Trade.Type == "Standard"), ]
levels(factor(dt2$Trade.Type))  #check Standard only
```

```
## [1] "Standard"
```

```r
dt2 = dt2[order(dt2$Seller.ID, dt2$Advance.Date), ]  #subset
```

Task 2

Add a couple of features to the dataset: a. The expected duration: the number of days between advance date and expected payment date. b. The number of previously settled trades for the seller. Make sure to only count trades where the Settlement Date is before the Advance Date of the trade.

Notes:

- Set the expected duration to zero if the expected payment date is less than the advance date
- Double sapply to create the settled trades - not elegant, probably a R package that can simplify
- Variant of settled trades to only incldues trades that are settled prior to or equal to the expected payment date

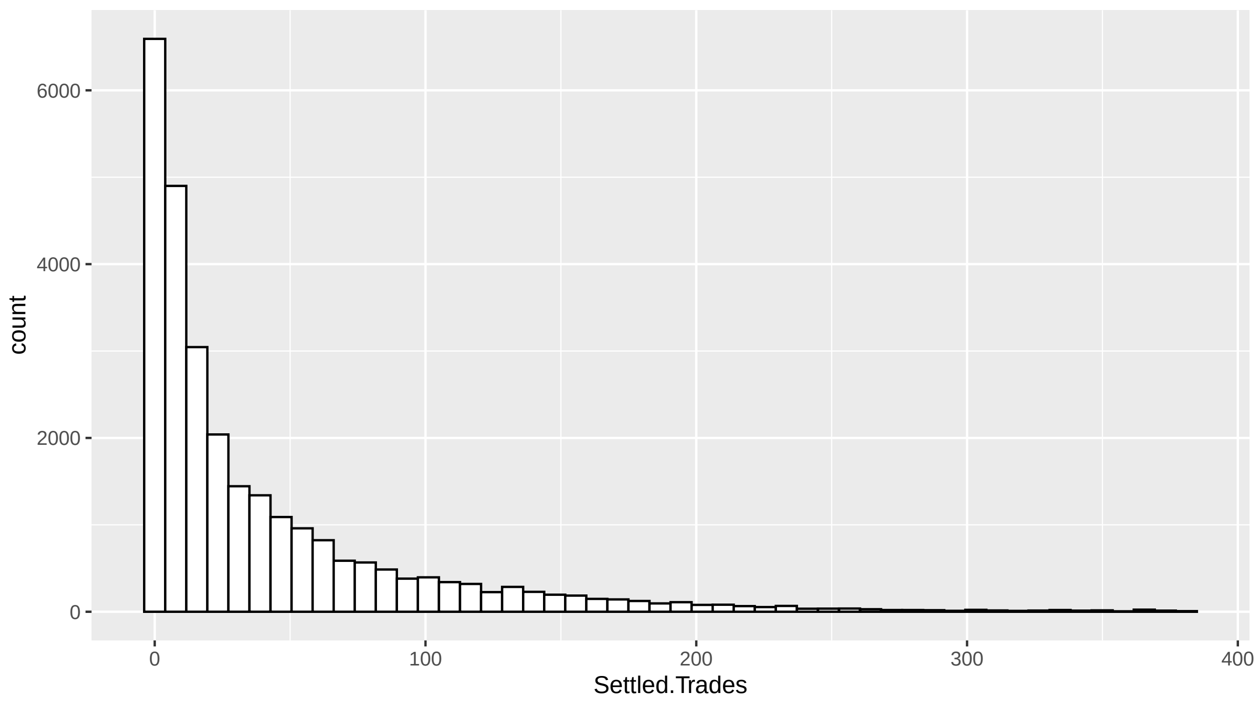
```r
# create features - expected duration
dt2$Expected.Duration = as.numeric(dt2$Expected.Payment.Date - dt2$Advance.Date)
levels(factor(dt2$Expected.Duration[dt2$Expected.Duration < 0]))
```

```
##  [1] "-285" "-278" "-62"  "-18"  "-17"  "-12"  "-10"  "-6"   "-5"   "-4"
## [11] "-3"   "-2"
```

```r
# View(dt2[dt2$Expected.Duration<0,]) #Expected Payment prior to Advance Date -
# set to zero
dt2$Expected.Duration = ifelse(dt2$Expected.Duration < 0, 0, dt2$Expected.Duration)
levels(factor(dt2$Expected.Duration[dt2$Expected.Duration < 0]))  #check
```

```
## character(0)
```

```r
# create features - settled trades
dt2$Settled.Ind = ifelse(!is.na(dt2$Settlement.Date), 1, 0)
dt2$Settled.Prior.Expected.Ind = ifelse(!is.na(dt2$Settlement.Date) & (dt2$Settlement.Date <=
    dt2$Expected.Payment.Date), 1, 0)
dt2$Settled.Trades = unlist(sapply(unique(dt2$Seller.ID), function(x) sapply(dt2$Advance.Date[dt2$Selle
    x], function(y) sum(dt2$Settled.Ind[dt2$Seller.ID == x & dt2$Settlement.Date <=
    y & !is.na(dt2$Settlement.Date)]))))

dt2$Settled.Trades.Prior.Expected = unlist(sapply(unique(dt2$Seller.ID), function(x) sapply(dt2$Advance
    x], function(y) sum(dt2$Settled.Prior.Expected.Ind[dt2$Seller.ID == x & dt2$Settlement.Date <=
    y & !is.na(dt2$Settlement.Date)]))))
# levels(factor(dt2$Settled.Trades))
ggplot(dt2, aes(x = Settled.Trades)) + geom_histogram(color = "black", fill = "white",
    bins = 50)  # Basic histogram
```



Task 3

Fit a logistic regression to predict In Arrears by using the two features that you created in the previous step. Interpret your findings and explain how you have evaluated the model performance.

Notes:

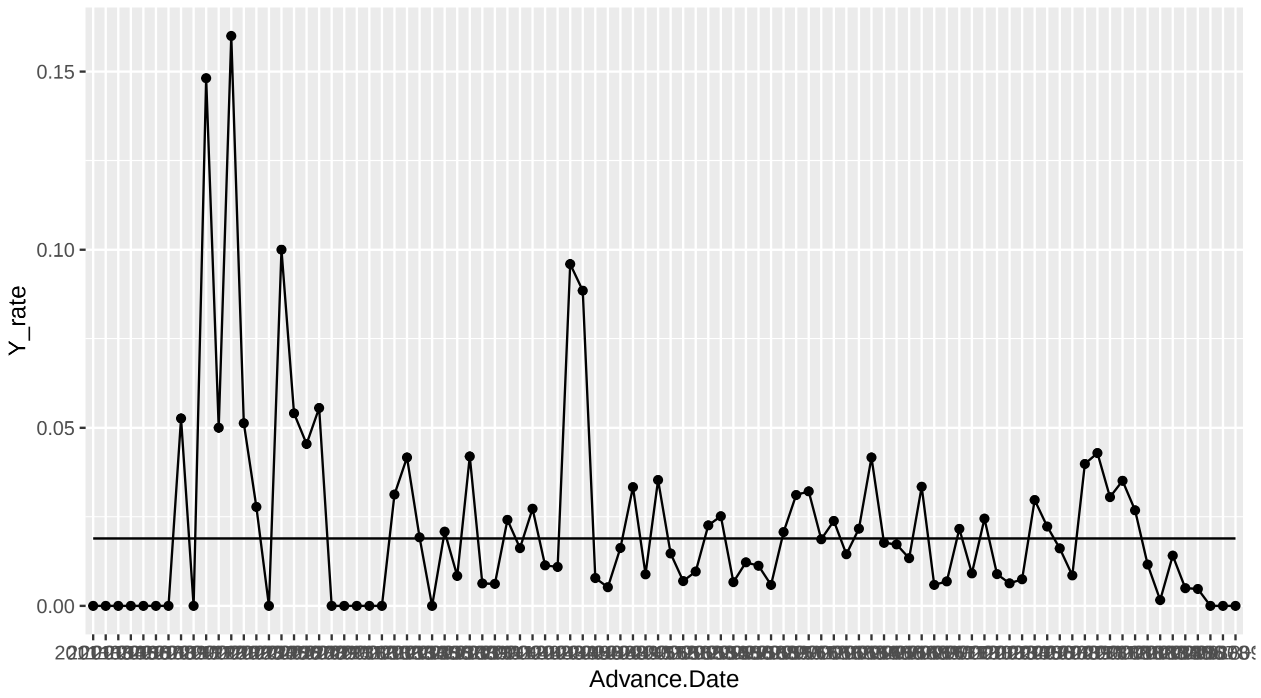
- Binary dependent variable calculated from In.Arrears ("Yes" = 1)
- Time series plot of dependent variable shows peaks that could indicate macro effects
- Logistic regression model is fit using glm/ glmnet (lambda=0) and model is assessed with k-fold (k=5) cross validation - model performance assessed using AUC / Gini
- Two features (Expected Duration and Settled Trades) results in a Test Gini of 21%
- Negative beta coefficient for Settled Trades. This is intuitive, higher settled trades implies lower probability of arrears. However, p-value indicates the feature is not statistically significant.
- Positive beta coefficient for Expected Duration This is also intuitive, higher durations implies higher probability of arrears.

```
# glm1 simple logistic with k-fold CV
levels(factor(dt2$In.Arrears))
```

```
## [1] "No"  "Yes"
```

```
Y = ifelse(dt2$In.Arrears == "Yes", 1, 0)  # define dependent variable
Y_hat = sum(Y)/length(Y)  #0.01890802
Y_ts = aggregate(Y, by = list(format(as.Date(dt2$Advance.Date), "%Y-%m")), sum)
lengthY_ts = aggregate(Y, by = list(format(as.Date(dt2$Advance.Date), "%Y-%m")),
    length)
```

```
ts = cbind(Y_ts, lengthY_ts[, 2], Y_hat)
colnames(ts) = c("Advance.Date", "Y_sum", "obs", "Y_hat")
ts$Y_rate = ts$Y_sum/ts$obs
ggplot(data = ts, aes(x = Advance.Date, y = Y_rate, group = 1)) + geom_line() + geom_point() +
    geom_line(data = ts, aes(x = Advance.Date, y = Y_hat, group = 1))  #check time series
```



```
X = dt2[, c("Expected.Duration", "Settled.Trades")]  # define feature(s)

# glm k-fold function
cv_glm_function_1 = function(X, Y, family, kfold, lambda = 0, alpha = 1) {
    st = Sys.time()
    kfold.list = groupKFold(group = dt2$Seller.ID, k = 5)

    cv_glm = function(k) {
        k.trn = kfold.list[[k]]
        glm.trn = glmnet(y = Y[k.trn], x = as.matrix(X[k.trn, ]), family = family,
            lambda = lambda, alpha = alpha)
        score = as.numeric(predict(glm.trn, newx = as.matrix(X), type = "response",
            exact = TRUE))
        # platt_coefs = glm(Y[k.trn] ~ score[k.trn], family = 'binomial')$coefficients #
        # platt scaling score = (1 / (1 + exp(-(platt_coefs[2] * score +
        # platt_coefs[1])))) # results are not as expected
        gini.trn = 2 * auc(response = Y[k.trn], predictor = score[k.trn], quiet = TRUE) -
            1
        gini.tst = 2 * auc(response = Y[-k.trn], predictor = score[-k.trn], quiet = TRUE) -
            1
        # roc.tst = plot.roc(x = Y[-k.trn], predictor = score[-k.trn], percent=TRUE)

        list(score = as.matrix(score), k = k, gini.trn = gini.trn, gini.tst = gini.tst,
            glm.a0 = glm.trn$a0, glm.beta = glm.trn$beta, glm.lambda = glm.trn$lambda)
    }

    list.cv = lapply(1:kfold, cv_glm)
    score = sapply(list.cv, "[[", "score")
    gini.tst = sapply(list.cv, "[[", "gini.tst")
    gini.trn = sapply(list.cv, "[[", "gini.trn")
    a0 = sapply(list.cv, "[[", "glm.a0")
    beta = sapply(list.cv, "[[", "glm.beta")
    lambda = sapply(list.cv, "[[", "glm.lambda")

    # plot.roc(x = Y, predictor = score[,1], percent=TRUE) for (i in 2:kfold)
    # {lines.roc(x= Y, predictor = score[,i], percent=TRUE)}
    print(paste0("Test Gini: ", mean(gini.tst)))
    print(paste0("Train Gini: ", mean(gini.trn)))

    list(score = score, gini.tst = gini.tst, gini.trn = gini.trn, a0 = a0, beta = beta,
        lambda = lambda)
}

glm = glm(formula = Y ~ as.matrix(X), family = "binomial")  # fit on full data
summary(glm)
```

```
##
## Call:
## glm(formula = Y ~ as.matrix(X), family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3677  -0.2053  -0.1844  -0.1700   3.0547
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  -4.5481989  0.0873338 -52.078  < 2e-16 ***
## as.matrix(X)Expected.Duration  0.0142733  0.0013359  10.685  < 2e-16 ***
## as.matrix(X)Settled.Trades    -0.0027389  0.0009201  -2.977  0.00291 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5206.6  on 27765  degrees of freedom
## Residual deviance: 5110.3  on 27763  degrees of freedom
## AIC: 5116.3
##
## Number of Fisher Scoring iterations: 7
```

```
glm1 = cv_glm_function_1(X = X, Y = Y, kfold = 5, family = "binomial")  # run simple GLM
```

```
## [1] "Test Gini: 0.190232825124713"
## [1] "Train Gini: 0.232794315733403"
```

```
# print(glm1$beta) # print beta coefficients
```

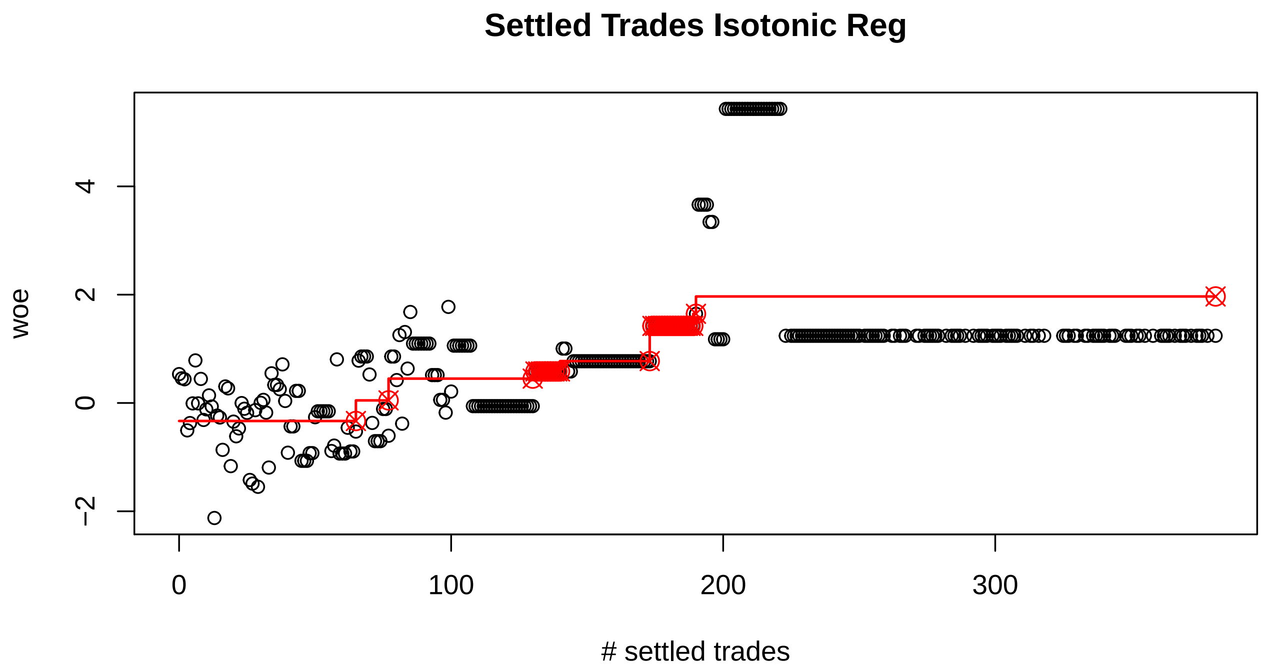
Task 4

Have a look at the relationship between previously settled trades and In Arrears. Can you visualise this relationship? How would you alter your model based on this? What is the impact on performance?

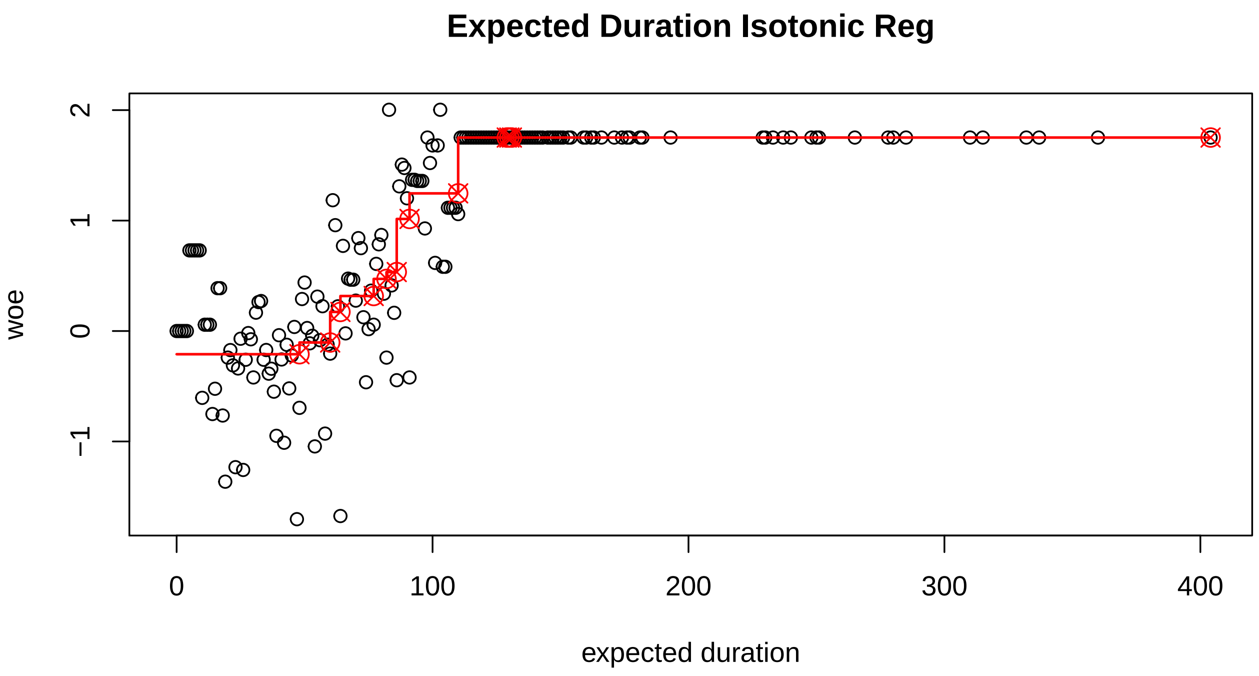
Notes:

- Weight of Evidence (WOE) and Isotonic Regression are used to explore relationship between features and dependent variable .
- WOE is natural logarithm of the proportion of Bads within a given group divided by the proportion of Goods within a given group; the greater the value of WOE, the higher the chance of observing arrears.
- Plot of settled trades and WOE is not intuitive - would expect the chance of arrears to decrease for increasing settled trades, however the WOE tends to be increasing for increasing settled trades.
- Plot also shows that the settled trades would benefit from grouping. Isotonic regression is investigated here to see if a simple, monotonic grouping would improve model performance. However, the GLM3 model fit, which uses the monotonic grouping, shows that this does not work well with a Test Gini of 13%. For comparison, the GLM2 model fit, which is the same as the two feature GLM1 model but with the WOE used for the dependent variables, results in a Test Gini of 39%.

```r
# Calculate WOE
weight.of.evidence = function(X, Y) {
    evidence = NULL
    woe = NULL
    weights = rep(1, length(Y))
    Y = Y * weights
    lengthY = sum(weights)
    prior = sum(Y)/lengthY

    for (name in colnames(X)) {
        col = factor(X[, name])
        colrslt = as.numeric(col)
        levelsCol = levels(col)
        prev.evdnc = 0
        for (lvl in levelsCol) {
            ttl = na.omit(Y[col == lvl])
            lengthttl = sum(na.omit(weights[col == lvl]))

            if (sum(ttl) < 1)
                evdnc = prev.evdnc  # at least one bad
 else if (lengthttl < 10)
                evdnc = prev.evdnc  # at least ten trades
 else if (lengthttl - sum(ttl) == 0)
                evdnc = 1 else evdnc = log(((sum(ttl))/sum(Y))/(((lengthttl - sum(ttl)))/(lengthY -
                sum(Y))))

            colrslt[col == lvl] = evdnc
            evidence = rbind(evidence, c(name, lvl, evdnc))
            prev.evdnc = evdnc
        }
        woe = cbind(woe, colrslt)
    }
    colnames(woe) = colnames(X)
    woe[is.na(woe)] = 0
    colnames(evidence) = c("colname", "level", "woe")
    list(woe = woe, evidence = evidence)
}

WOE1 = weight.of.evidence(X = X, Y = Y)  # calculate WOE
WOE.Settled.Trades = na.omit(WOE1$evidence[which(WOE1$evidence[, 1] == "Settled.Trades"),
    ])  # WOE for settled trades
Settled.Trades.Monotonic = isoreg(x = WOE.Settled.Trades[, 2], y = WOE.Settled.Trades[,
    3])  # fit isotonic regression, output is a monotonic grouped WOE
plot(Settled.Trades.Monotonic, main = "Settled Trades Isotonic Reg", xlab = "# settled trades",
    ylab = "woe")  # plot to see isotonic reg fit
```



```
Settled.Trades.Grouped = as.data.frame(cbind(Settled.Trades.Monotonic$x, Settled.Trades.Monotonic$y,
    Settled.Trades.Monotonic$yf))  # create adjusted settled trade feature - WOE and monotonic grouped
colnames(Settled.Trades.Grouped) = c("Settled.Trades", "Settled.Trades.WOE", "Settled.Trades.Grouped")

WOE.Expected.Duration = na.omit(WOE1$evidence[which(WOE1$evidence[, 1] == "Expected.Duration"),
    ])  # WOE for expected duration
Expected.Duration.Monotonic = isoreg(x = WOE.Expected.Duration[, 2], y = WOE.Expected.Duration[,
    3])  # fit isotonic regression, output is a monotonic grouped WOE
plot(Expected.Duration.Monotonic, main = "Expected Duration Isotonic Reg", xlab = "expected duration",
    ylab = "woe")  # plot to see isotonic reg fit
```



```
Expected.Duration.Grouped = as.data.frame(cbind(Expected.Duration.Monotonic$x, Expected.Duration.Monotor
    Expected.Duration.Monotonic$yf))  # create adjusted expected duration feature - WOE and monotonic g
colnames(Expected.Duration.Grouped) = c("Expected.Duration", "Expected.Duration.WOE",
    "Expected.Duration.Grouped")

X2 = join(x = X, y = Expected.Duration.Grouped, by = "Expected.Duration", type = "left",
    match = "all")
X2 = join(x = X2, y = Settled.Trades.Grouped, by = "Settled.Trades", type = "left",
    match = "all")

# glm2 with WOE
glm = glm(formula = Y ~ as.matrix(X2[, c(3, 5)]), family = "binomial")  # fit on full data with settled
summary(glm)
```

```
##
## Call:
## glm(formula = Y ~ as.matrix(X2[, c(3, 5)]), family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0638  -0.2012  -0.1692  -0.1387   3.4183
##
## Coefficients:
##                                              Estimate Std. Error z value
## (Intercept)                                  -4.15154    0.05108  -81.28
## as.matrix(X2[, c(3, 5)])Expected.Duration.WOE  0.87571    0.06203   14.12
## as.matrix(X2[, c(3, 5)])Settled.Trades.WOE     0.47118    0.03271   14.40
##                                              Pr(>|z|)
## (Intercept)                                    <2e-16 ***
## as.matrix(X2[, c(3, 5)])Expected.Duration.WOE   <2e-16 ***
## as.matrix(X2[, c(3, 5)])Settled.Trades.WOE      <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5206.6  on 27765  degrees of freedom
## Residual deviance: 4869.7  on 27763  degrees of freedom
## AIC: 4875.7
##
## Number of Fisher Scoring iterations: 7
```

```
glm2 = cv_glm_function_1(X = X2[, c(3, 5)], Y = Y, kfold = 5, family = "binomial")  #0.3819618
```

```
## [1] "Test Gini: 0.387564259904301"
## [1] "Train Gini: 0.397790035090183"
```

```
# print(glm2$beta)

# glm3 with monotonic grouping - doesn't improve model perf
glm = glm(formula = Y ~ as.matrix(X2[, c(4, 6)]), family = "binomial")  # fit on full data with settled
summary(glm)
```

```
##
## Call:
## glm(formula = Y ~ as.matrix(X2[, c(4, 6)]), family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.4833  -0.1888  -0.1724  -0.1724   2.9037
##
## Coefficients:
##                                                     Estimate Std. Error z value
## (Intercept)                                         -3.96048    0.04576 -86.557
## as.matrix(X2[, c(4, 6)])Expected.Duration.Grouped  0.88353    0.08231  10.734
## as.matrix(X2[, c(4, 6)])Settled.Trades.Grouped     0.16487    0.08009   2.059
##                                                     Pr(>|z|)
## (Intercept)                                           <2e-16 ***
## as.matrix(X2[, c(4, 6)])Expected.Duration.Grouped   <2e-16 ***
## as.matrix(X2[, c(4, 6)])Settled.Trades.Grouped      0.0395 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5206.6  on 27765  degrees of freedom
## Residual deviance: 5109.2  on 27763  degrees of freedom
## AIC: 5115.2
##
## Number of Fisher Scoring iterations: 7
```

```
glm3 = cv_glm_function_1(X = X2[, c(4, 6)], Y = Y, kfold = 5, family = "binomial")  #0.07532676
```

```
## [1] "Test Gini: 0.145142926719734"
## [1] "Train Gini: 0.219772582762502"
```

```
# print(glm3$beta)
```

Task 5

Try to improve the model performance metric. What other variables could be predictive? What other modelling techniques could you try? Provide brief comments on why you decided to work on a particular idea.

Notes:

- Additional features can be considered. Three additional features have been investigated below - Price Grade, Face Value GBP and Advance Value GBP.
- Coarse classing/binning algorithm based on the WOE can be used to reduce noise
- Feature selection algorithms that maximise the performance metric (AUC/Gini) can be considered, such as Information Value ordering, Stepwise selection algorithms, Regularisation (Ridge/Lasso/Elastic-net). Lasso is investigated below as a form of feature selection.
- Refinement to the dependent variable can be considered, such as the number of days in Arrears (grouped or continious) or the value of Arrears (grouped or continuous). This is dependent on the business case/ use of the model.
- Other classification models (Random Forests or Mixed models) or optimisation algorithms (Boosting) can be considered. The data can be considered panel data (Seller can be measured repeatedly) and therefore a mixed effect model may be useful, for example, to isolate the macro effect from the idiosyncratic effects or model variance within groups of Sellers.
- The GLM4 model fit adds the Price Grade, Face Value GBP, Advance Value GBP and Settled Trades Prior to Expected. The result is a Test Gini of 47%. The p-value for the Face Value GBP and Advance Value GBP indicate the features are not statistically significant.
- The GLM5 model fit adds Lasso (with AUC optimised lambda value selected using the cv.glmnet function). The result is a Test Gini of 46% and both the Face Value GBP and Advance Value GBP beta coefficients are shrunk to zero.
- Finally, the ROC curve compares the 5 GLM model fits in this report - the GLM5 model fit (4 features with WOE) is the proposed model.

```r
# glm4 additional features
X3 = dt2[, c("Price.Grade", "Face.Value..GBP.", "Advance..GBP.", "Settled.Trades",
    "Settled.Trades.Prior.Expected", "Expected.Duration")]
WOE2 = weight.of.evidence(X = X3, Y = Y)
glm = glm(formula = Y ~ as.matrix(WOE2$woe), family = "binomial")  # fit on full data
summary(glm)
```

```
##
## Call:
## glm(formula = Y ~ as.matrix(WOE2$woe), family = "binomial")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0517  -0.2123  -0.1496  -0.1025   3.7179
##
## Coefficients:
##                                                   Estimate Std. Error z value
## (Intercept)                                       -3.98247    0.12439 -32.016
## as.matrix(WOE2$woe)Price.Grade                     1.07515    0.08096  13.280
## as.matrix(WOE2$woe)Face.Value..GBP.                0.02480    0.05588   0.444
## as.matrix(WOE2$woe)Advance..GBP.                  -0.19220    0.08579  -2.240
## as.matrix(WOE2$woe)Settled.Trades                  0.37479    0.04192   8.942
## as.matrix(WOE2$woe)Settled.Trades.Prior.Expected  0.31843    0.04642   6.860
## as.matrix(WOE2$woe)Expected.Duration              0.88901    0.06337  14.030
##                                                  Pr(>|z|)
## (Intercept)                                       < 2e-16 ***
## as.matrix(WOE2$woe)Price.Grade                    < 2e-16 ***
## as.matrix(WOE2$woe)Face.Value..GBP.                0.6572
## as.matrix(WOE2$woe)Advance..GBP.                   0.0251 *
## as.matrix(WOE2$woe)Settled.Trades                 < 2e-16 ***
## as.matrix(WOE2$woe)Settled.Trades.Prior.Expected 6.91e-12 ***
## as.matrix(WOE2$woe)Expected.Duration              < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5206.6  on 27765  degrees of freedom
## Residual deviance: 4629.0  on 27759  degrees of freedom
## AIC: 4643
##
## Number of Fisher Scoring iterations: 7
```

```
glm4 = cv_glm_function_1(X = WOE2$woe, Y = Y, kfold = 5, family = "binomial")  #0.4614117
```

```
## [1] "Test Gini: 0.470752951599822"
## [1] "Train Gini: 0.486097132975904"
```

```
# glm5 lasso feature selection
glm.lambda.opt = cv.glmnet(y = Y, x = as.matrix(WOE2$woe), family = "binomial", type.measure = "auc",
    nfolds = 5, alpha = 1)
glm.lambda = glmnet(y = Y, x = as.matrix(WOE2$woe), family = "binomial", type.measure = "auc",
    nfolds = 5, lambda = glm.lambda.opt$lambda.1se, alpha = 1)  #s0
glm.lambda$beta  #drop Face.Value.GBP and Advance.GBP
```

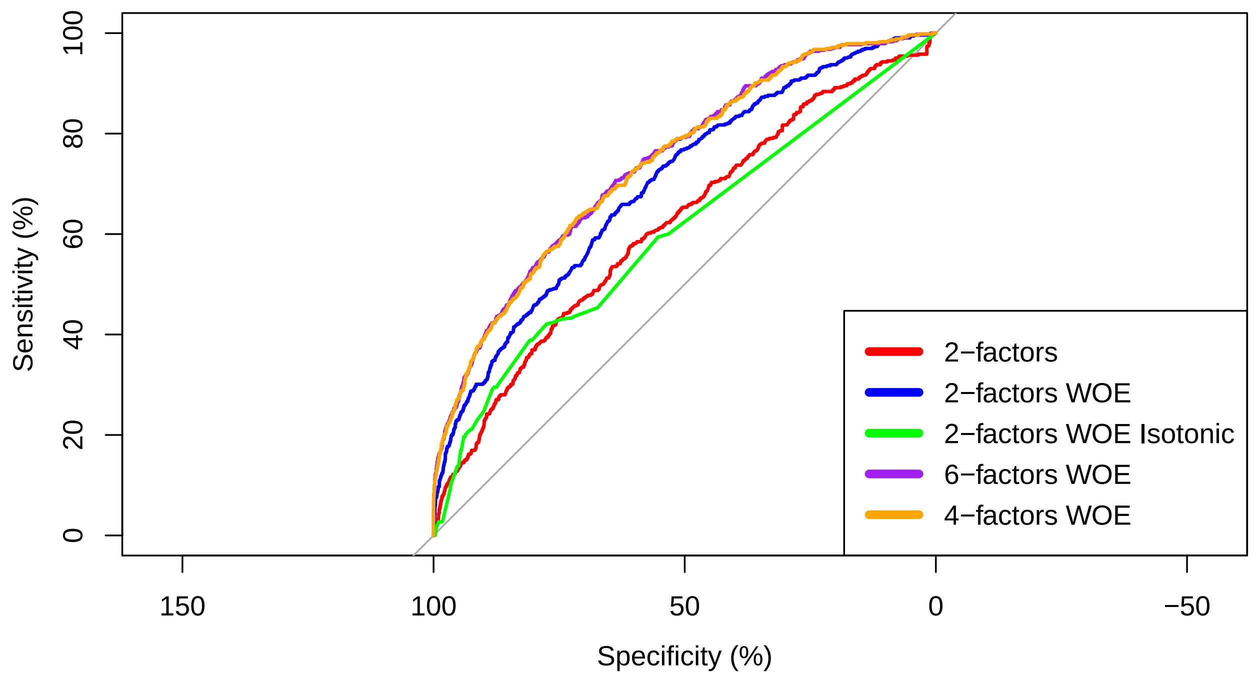
```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                                        s0
## Price.Grade                    0.13522065
## Face.Value..GBP.               .
## Advance..GBP.                  .
## Settled.Trades                 0.15389865
## Settled.Trades.Prior.Expected 0.02410259
## Expected.Duration              0.20307525
```

```
X4 = dt2[, c("Price.Grade", "Settled.Trades", "Settled.Trades.Prior.Expected", "Expected.Duration")]
WOE3 = weight.of.evidence(X = X4, Y = Y)
glm5 = cv_glm_function_1(X = WOE3$woe, Y = Y, kfold = 5, family = "binomial")  #0.460982
```

```
## [1] "Test Gini: 0.470348424915664"
## [1] "Train Gini: 0.48538843614048"
```

```
# Compare GLM model roc curves
roc1 = plot.roc(x = Y, predictor = rowMeans(unlist(glm1$score)), percent = TRUE,
    col = "red", quiet = TRUE)
roc2 = lines.roc(x = Y, predictor = rowMeans(unlist(glm2$score)), percent = TRUE,
    col = "blue", quiet = TRUE)
roc3 = lines.roc(x = Y, predictor = rowMeans(unlist(glm3$score)), percent = TRUE,
    col = "green", quiet = TRUE)
```

```
roc4 = lines.roc(x = Y, predictor = rowMeans(unlist(glm4$score)), percent = TRUE,
    col = "purple", quiet = TRUE)
roc5 = lines.roc(x = Y, predictor = rowMeans(unlist(glm5$score)), percent = TRUE,
    col = "orange", quiet = TRUE)
legend("bottomright", legend = c("2-factors", "2-factors WOE", "2-factors WOE Isotonic",
    "6-factors WOE", "4-factors WOE"), col = c("red", "blue", "green", "purple",
    "orange"), lwd = 5)
```


Sensitivity (%)
Specificity (%)
2−factors
2−factors WOE
2−factors WOE Isotonic
6−factors WOE
4−factors WOE