CLAREMONT McKENNA COLLEGE

Predicting Corporate Bankruptcy Using KNN and Logistic Regression

Umut Araç, Jake Norville, and Carsten Savage
Professor Batta
ECON160: Accounting Data Analytics
14 May 2021

Project Overview

The goal of this project was to employ R Studio and machine learning algorithms to predict whether a given firm will go bankrupt based on financial ratios.

There were two datasets we used for the project. The first consisted of financial ratios for firm-years and was extracted from Compustat. The second dataset consisted of the dates that firms declared bankruptcy and was extracted from the Audit Analytics database through WRDS.

We first selected the financial ratios that we believed are most indicative of bankruptcy risk, and we made these our predictor variables. The ratios that we selected are the following: (1) Cash Flow/Total Debt, (2) Current Liabilities/Total Liabilities, (3) Interest/Average Total Debt, (4) Quick Ratio, (5) Total Debt/Equity, (6) Total Debt/Total Assets, and (7) After-Tax Interest Coverage.

We decided to use the unaltered financial ratios for our predictor variables rather than the changes in ratios or the lagged values. This is because the unaltered financial ratios likely more accurately gauge firm performance and are easily comparable across firms and industries.

After merging the datasets, we "forward-filled" the rows for bankrupt firm-years when the date of that company's latest financial statement was one to four years before the date that the company declared bankruptcy. If the firm ceased issuing financial statements in 2016, for example, and the firm declared bankruptcy in 2019, we forward-filled that firm's rows to include 2017, 2018, and 2019.

Once we forward-filled firm-years, we created a binary outcome variable, 'isBankrupt,' which equals 1 if the firm went bankrupt and 0 if the firm did not go bankrupt. A binary outcome variable is essential for both logit and KNN regressions.

We winsorized and computed summary statistics for the bankrupt and non-bankrupt firms and created a frequency table displaying the number of firm bankruptcies and non-bankruptcies in each year.

To predict firm bankruptcies, we employed the 'boot' and 'caret' packages in R. We winsorized the dataset for the logit model and ran a logit regression that used polynomial terms ranging from 1 to 5. We then computed the overall error and model error costs for the different specifications of the logit model by iterating over the number of folds and the number of polynomials in 5- and 10-fold cross validation.

We scaled the dataset for the KNN model and ran K-nearest neighbors for 1 to 5 near neighbors in different specifications, iterating over the number of near neighbors and the number of folds in 5- and 10-fold cross validation. To compute the model error costs for KNN, we created a vector consisting of a combination of model sensitivity, specificity, and the costs of false negatives and false positives.

According to our table, which includes overall error and model error costs for both the logit and KNN models, the overall error for the logit model is lower than that of the KNN model, but overall errors for both models are relatively similar and are within .00469 to .01029. The optimal logit model appears to have a polynomial term greater than 1 in 10-fold cross validation, which minimizes the overall error for 10-fold cross validation. The overall error for 5-fold cross validation was constant regardless of the polynomial term. The optimal KNN model appears to be K=5 near neighbors in 5-fold cross validation, which minimizes the overall error rate while having a lower model error cost than the second-best KNN model, which was K=5 near neighbors in 10-fold cross validation. The model error costs for both the logit and KNN models are also relatively similar and are within .001959 to .0025.

Bankruptcy

Umut, Jake, Carsten

4/24/2021

```r
library(knitr)
hook_output = knit_hooks$get('output')
knit_hooks$set(output = function(x, options) {
  # this hook is used only when the linewidth option is not NULL
  if (!is.null(n <- options$linewidth)) {
    x = knitr:::split_lines(x)
    # any lines wider than n should be wrapped
    if (any(nchar(x) > n)) x = strwrap(x, width = n)
    x = paste(x, collapse = '\n')
  }
  hook_output(x, options)
})
```

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(readxl)
library(DescTools)
library(ggrepel)
library(viridis)
```

```
## Loading required package: viridisLite
```

```r
library(stringr)
library(dplyr)
library(qwraps2)

getwd()
```

```
## [1] "/Users/carstenjuliansavage/Desktop/R Working Directory/Accounting/Accounting Data Analytics/Data
```

```r
setwd(str_c(getwd(),'/data/'))

BankruptciesCSV <- as_tibble(read.csv("bankruptcies.csv"))
Ratiosbyyear <- as_tibble(read.csv("ratios_by_year.csv"))
```

```r
# Selecting the variables we want to use for corporate bankruptcy prediction

Ratios <- Ratiosbyyear  %>%
  select(gvkey:public_date, cash_debt, curr_debt, int_totdebt, quick_ratio, de_ratio, debt_assets, intco
```

Here are the predictor variables we chose to include in our model and our reasoning: 1. Cash Flow/Total Debt – Indicates how long it would take a company to repay total debt if it only used its cash flow to repay the debt. Important measure of solvency, higher ratio suggests better overall financial health (Investopedia). 2. Current Liabilities/Total Liabilities – The proportion of total liabilities that are due within the year/short term. A higher ratio indicates that a company has more debt burden in the short term and may be at a greater risk of bankruptcy. 3. Interest/Average Total Debt – Measures the cost of a company's debt. A higher ratio indicates a larger debt burden and could indicate that a company is at greater risk of bankruptcy because it cannot get low-interest loans from the bank. 4. Quick Ratio – Measures whether the liquid assets of the company are sufficient to cover current liabilities in the short term. A lower ratio indicates that a company would be at a greater risk of insolvency/bankruptcy than a company with a higher quick ratio. 5. Total Debt/Equity – It suggests whether shareholders' equity can cover company debt. Measures indebtedness and can indicate insolvency. A higher total debt/equity ratio indicates higher leverage and higher risk to shareholders. (Investopedia).
6. Total Debt/Total Assets – A higher ratio indicates higher leverage and also higher risk, suggesting that companies with higher total-debt-to-total-asset ratios are at higher risk of insolvency/bankruptcy (Investopedia).
7. After-Tax Interest Coverage (EBIT/Interest Expense) – A higher ratio indicates that a company is less burdened by interest expense and is more solvent.

```r
options(scipen=999)

DLTA <- function(x) {
(x-lag(x)) / lag(x)
}

#Add more predictors here to make them into lagged vars
Ratiosbyyear <- Ratiosbyyear %>%
  mutate(across(c(cash_debt, curr_debt, int_totdebt, quick_ratio, de_ratio, debt_assets, intcov),list(dl

is.na(Ratiosbyyear) <- Ratiosbyyear %>%
  sapply(is.infinite)
```

```r
Ratiosbyyear <- Ratiosbyyear %>%
  drop_na(cash_debt_dl:intcov_dl)
```

```r
# Making sure the dates belong to the Date class in both data sets (Filter 1988)
Ratiosbyyear  <- Ratiosbyyear %>%
  mutate(adate = ymd(adate),
         public_date = ymd(adate),
         adate_year = as.numeric(Year(adate)),
         public_year = as.numeric(Year(public_date)))

BankruptciesCSV <- BankruptciesCSV %>%
  mutate(bank_event = ymd(bank_event),
         bankruptcy_year = as.numeric(Year(bank_event)))

MasterData <- Ratiosbyyear %>%
  left_join(BankruptciesCSV) %>%
  as_tibble() %>%
  filter(adate_year >= 1988)
```

```
## Joining, by = "COMPANY_FKEY"
```

```r
Y12df <- MasterData %>%
  group_by(COMPANY_FKEY) %>%
  filter(!is.na(bank_event) & adate==max(adate)) %>%
  filter(bank_event-adate > 365 & bank_event-adate <= 365*2) %>%
  mutate(Added=NA_integer_) %>%

    group_by(COMPANY_FKEY) %>%
    # added=1 (or 1:1) will add one row, added=1:2 two rows, etc.
    do(add_row(.,Added=1:1)) %>%
    ungroup() %>%

    #fill() fills in NA values with prior row values that aren't NA. Here, I've filled in everything bu
    fill(-c(Added)) %>%
    group_by(COMPANY_FKEY) %>%

    # This lets me add 1 to the year variable, based on the within-name row_number
    mutate(adate_year = adate_year + row_number()-1) %>%
    mutate(adate = adate %m+% years(1)) %>%
    ungroup() %>%

    # dump all but the newly-added rows
    filter(is.na(Added)==FALSE)

Y12df
```

```
## # A tibble: 302 x 91
##    gvkey permno adate      public_date   CAPEI     bm     evm pe_op_basic
##    <int>  <int> <date>     <date>        <dbl>  <dbl>   <dbl>       <dbl>
##  1  1278  77829 2000-12-31 1999-12-31   -0.21  NA      32.4        NA
##  2 63192  83719 1999-12-31 1998-12-31   -2.48   0.9    20.9        -1.34
##  3  2033  29532 2008-09-30 2007-09-30   -1.80   2.03   -2.08       -1.34
```

```
## 4 2215 55044 2000-11-30 1999-11-30 61.9 0.357 10.5 10.2
## 5 2393 17961 2020-06-30 2019-06-30 -46.0 3.12 30.3 10.2
## 6 2625 20723 2001-06-30 2000-06-30 -3.22 0.408 -31.5 10.2
## 7 2811 30402 2016-12-31 2015-12-31 -0.109 1.35 -51.1 -0.071
## 8 3156 64901 2008-12-31 2007-12-31 -0.086 1.35 -0.876 -0.056
## 9 2555 20248 2013-01-31 2012-01-31 -1.03 1.35 6.18 -0.219
## 10 3833 60273 2010-12-31 2009-12-31 -3.23 2.24 -4.56 -2.24
## # ... with 292 more rows, and 83 more variables: pe_op_dil <dbl>, pe_exi <dbl>,
## #   pe_inc <dbl>, ps <dbl>, pcf <dbl>, dpr <dbl>, npm <dbl>, opmbd <dbl>,
## #   opmad <dbl>, gpm <dbl>, ptpm <dbl>, cfm <dbl>, roa <dbl>, roe <dbl>,
## #   roce <dbl>, efftax <dbl>, aftret_eq <dbl>, aftret_invcapx <dbl>,
## #   aftret_equity <dbl>, pretret_noa <dbl>, pretret_earnat <dbl>, GProf <dbl>,
## #   equity_invcap <dbl>, debt_invcap <dbl>, totdebt_invcap <dbl>,
## #   capital_ratio <dbl>, int_debt <dbl>, int_totdebt <dbl>, cash_lt <dbl>,
## #   invt_act <dbl>, rect_act <dbl>, debt_at <dbl>, debt_ebitda <dbl>,
## #   short_debt <dbl>, curr_debt <dbl>, lt_debt <dbl>, profit_lct <dbl>,
## #   ocf_lct <dbl>, cash_debt <dbl>, fcf_ocf <dbl>, lt_ppent <dbl>,
## #   dltt_be <dbl>, debt_assets <dbl>, debt_capital <dbl>, de_ratio <dbl>,
## #   intcov <dbl>, intcov_ratio <dbl>, cash_ratio <dbl>, quick_ratio <dbl>,
## #   curr_ratio <dbl>, cash_conversion <dbl>, inv_turn <dbl>, at_turn <dbl>,
## #   rect_turn <dbl>, pay_turn <dbl>, sale_invcap <dbl>, sale_equity <dbl>,
## #   sale_nwc <dbl>, rd_sale <dbl>, adv_sale <dbl>, staff_sale <dbl>,
## #   accrual <dbl>, ptb <dbl>, PEG_trailing <dbl>, divyield <chr>, TICKER <chr>,
## #   cusip <chr>, fyear <int>, datadate <int>, tic <chr>, COMPANY_FKEY <int>,
## #   cash_debt_dl <dbl>, curr_debt_dl <dbl>, int_totdebt_dl <dbl>,
## #   quick_ratio_dl <dbl>, de_ratio_dl <dbl>, debt_assets_dl <dbl>,
## #   intcov_dl <dbl>, adate_year <dbl>, public_year <dbl>, bank_event <date>,
## #   bankruptcy_year <dbl>, Added <int>
```

```
Y23df <-MasterData %>%
  group_by(COMPANY_FKEY) %>%
  filter(!is.na(bank_event) & adate==max(adate)) %>%
  filter(bank_event-adate>365*2 & bank_event-adate <=365*3) %>%
  mutate(Added=NA_integer_) %>%
    group_by(COMPANY_FKEY) %>%

    # added=1 (or 1:1) will add one row, added=1:2 two rows, etc.
    do(add_row(.,Added=1:2)) %>%
    ungroup() %>%

    #fill() fills in NA values with prior row values that aren't NA. Here, I've filled in everything bu
    fill(-c(Added)) %>%
    group_by(COMPANY_FKEY) %>%

    # This lets me add 1 to the year variable, based on the within-name row_number
    mutate(adate_year = adate_year+row_number()-1) %>%
    mutate(adate = if_else(Added == 1,adate %m+% years(1),adate %m+% years(2))) %>%
    ungroup() %>%

    # dump all but the newly-added rows
    filter(is.na(Added)==FALSE)

Y23df
```

```
## # A tibble: 212 x 91
##    gvkey permno adate      public_date  CAPEI     bm   evm pe_op_basic pe_op_dil
##    <int>  <int> <date>     <date>       <dbl>  <dbl> <dbl>       <dbl>     <dbl>
##  1  1627  83841 2000-09-30 1999-09-30  -0.037 NA     10.0       -0.036    -0.036
##  2  1627  83841 2001-09-30 1999-09-30  -0.037 NA     10.0       -0.036    -0.036
##  3  1372  12320 2003-08-31 2002-08-31  -7.33   3.74   9.16      -2.38     -2.38
##  4  1372  12320 2004-08-31 2002-08-31  -7.33   3.74   9.16      -2.38     -2.38
##  5  4412  80119 2000-12-31 1999-12-31  -0.668  3.74   7.81      -1.06     -1.06
##  6  4412  80119 2001-12-31 1999-12-31  -0.668  3.74   7.81      -1.06     -1.06
##  7  5043  62607 2001-09-30 2000-09-30  -0.12   0.021 -3.38      -0.855    -0.855
##  8  5043  62607 2002-09-30 2000-09-30  -0.12   0.021 -3.38      -0.855    -0.855
##  9  5508  79837 2000-09-30 1999-09-30  -0.134  1.79   3.62      -0.309    -0.309
## 10  5508  79837 2001-09-30 1999-09-30  -0.134  1.79   3.62      -0.309    -0.309
## # ... with 202 more rows, and 82 more variables: pe_exi <dbl>, pe_inc <dbl>,
## #   ps <dbl>, pcf <dbl>, dpr <dbl>, npm <dbl>, opmbd <dbl>, opmad <dbl>,
## #   gpm <dbl>, ptpm <dbl>, cfm <dbl>, roa <dbl>, roe <dbl>, roce <dbl>,
## #   efftax <dbl>, aftret_eq <dbl>, aftret_invcapx <dbl>, aftret_equity <dbl>,
## #   pretret_noa <dbl>, pretret_earnat <dbl>, GProf <dbl>, equity_invcap <dbl>,
## #   debt_invcap <dbl>, totdebt_invcap <dbl>, capital_ratio <dbl>,
## #   int_debt <dbl>, int_totdebt <dbl>, cash_lt <dbl>, invt_act <dbl>,
## #   rect_act <dbl>, debt_at <dbl>, debt_ebitda <dbl>, short_debt <dbl>,
## #   curr_debt <dbl>, lt_debt <dbl>, profit_lct <dbl>, ocf_lct <dbl>,
## #   cash_debt <dbl>, fcf_ocf <dbl>, lt_ppent <dbl>, dltt_be <dbl>,
## #   debt_assets <dbl>, debt_capital <dbl>, de_ratio <dbl>, intcov <dbl>,
## #   intcov_ratio <dbl>, cash_ratio <dbl>, quick_ratio <dbl>, curr_ratio <dbl>,
## #   cash_conversion <dbl>, inv_turn <dbl>, at_turn <dbl>, rect_turn <dbl>,
## #   pay_turn <dbl>, sale_invcap <dbl>, sale_equity <dbl>, sale_nwc <dbl>,
## #   rd_sale <dbl>, adv_sale <dbl>, staff_sale <dbl>, accrual <dbl>, ptb <dbl>,
## #   PEG_trailing <dbl>, divyield <chr>, TICKER <chr>, cusip <chr>, fyear <int>,
## #   datadate <int>, tic <chr>, COMPANY_FKEY <int>, cash_debt_dl <dbl>,
## #   curr_debt_dl <dbl>, int_totdebt_dl <dbl>, quick_ratio_dl <dbl>,
## #   de_ratio_dl <dbl>, debt_assets_dl <dbl>, intcov_dl <dbl>, adate_year <dbl>,
## #   public_year <dbl>, bank_event <date>, bankruptcy_year <dbl>, Added <int>
```

```r
Y34df <- MasterData %>%
  group_by(COMPANY_FKEY) %>%
  filter(!is.na(bank_event) & adate==max(adate)) %>%
  filter(bank_event-adate > 365*3 & bank_event-adate <= 365*4) %>%
  mutate(Added=NA_integer_) %>%
    group_by(COMPANY_FKEY) %>%

    # added=1 (or 1:1) will add one row, added=1:2 two rows, etc.
    do(add_row(.,Added=1:3)) %>%
    ungroup() %>%

    #fill() fills in NA values with prior row values that aren't NA. Here, I've filled in everything bu
    fill(-c(Added)) %>%
    group_by(COMPANY_FKEY) %>%

    # This lets me add 1 to the year variable, based on the within-name row_number
    mutate(adate_year= adate_year+row_number()-1) %>%
    mutate(adate = if_else(Added == 1,adate %m+% years(1),
                        if_else(Added == 2, adate %m+% years(2),adate %m+% years(3)))) %>%
    ungroup() %>%
```

```
    # dump all but the newly-added rows
    filter(is.na(Added)==FALSE)

Y34df
```

```
## # A tibble: 198 x 91
##    gvkey permno adate      public_date  CAPEI     bm     evm pe_op_basic
##    <int>  <int> <date>     <date>       <dbl>  <dbl>   <dbl>       <dbl>
##  1 12118  50657 1998-12-31 1997-12-31  -0.424 NA        4.32      -0.367
##  2 12118  50657 1999-12-31 1997-12-31  -0.424 NA        4.32      -0.367
##  3 12118  50657 2000-12-31 1997-12-31  -0.424 NA        4.32      -0.367
##  4  8390  79105 1998-01-31 1997-01-31  17.0    0.301    7.96       7.53
##  5  8390  79105 1999-01-31 1997-01-31  17.0    0.301    7.96       7.53
##  6  8390  79105 2000-01-31 1997-01-31  17.0    0.301    7.96       7.53
##  7  6424  19975 2001-12-31 2000-12-31  -1.77    1.86  -215.       -0.246
##  8  6424  19975 2002-12-31 2000-12-31  -1.77    1.86  -215.       -0.246
##  9  6424  19975 2003-12-31 2000-12-31  -1.77    1.86  -215.       -0.246
## 10  6548  48363 1998-12-31 1997-12-31   1.27    0.915   11.4      -0.446
## # ... with 188 more rows, and 83 more variables: pe_op_dil <dbl>, pe_exi <dbl>,
## #   pe_inc <dbl>, ps <dbl>, pcf <dbl>, dpr <dbl>, npm <dbl>, opmbd <dbl>,
## #   opmad <dbl>, gpm <dbl>, ptpm <dbl>, cfm <dbl>, roa <dbl>, roe <dbl>,
## #   roce <dbl>, efftax <dbl>, aftret_eq <dbl>, aftret_invcapx <dbl>,
## #   aftret_equity <dbl>, pretret_noa <dbl>, pretret_earnat <dbl>, GProf <dbl>,
## #   equity_invcap <dbl>, debt_invcap <dbl>, totdebt_invcap <dbl>,
## #   capital_ratio <dbl>, int_debt <dbl>, int_totdebt <dbl>, cash_lt <dbl>,
## #   invt_act <dbl>, rect_act <dbl>, debt_at <dbl>, debt_ebitda <dbl>,
## #   short_debt <dbl>, curr_debt <dbl>, lt_debt <dbl>, profit_lct <dbl>,
## #   ocf_lct <dbl>, cash_debt <dbl>, fcf_ocf <dbl>, lt_ppent <dbl>,
## #   dltt_be <dbl>, debt_assets <dbl>, debt_capital <dbl>, de_ratio <dbl>,
## #   intcov <dbl>, intcov_ratio <dbl>, cash_ratio <dbl>, quick_ratio <dbl>,
## #   curr_ratio <dbl>, cash_conversion <dbl>, inv_turn <dbl>, at_turn <dbl>,
## #   rect_turn <dbl>, pay_turn <dbl>, sale_invcap <dbl>, sale_equity <dbl>,
## #   sale_nwc <dbl>, rd_sale <dbl>, adv_sale <dbl>, staff_sale <dbl>,
## #   accrual <dbl>, ptb <dbl>, PEG_trailing <dbl>, divyield <chr>, TICKER <chr>,
## #   cusip <chr>, fyear <int>, datadate <int>, tic <chr>, COMPANY_FKEY <int>,
## #   cash_debt_dl <dbl>, curr_debt_dl <dbl>, int_totdebt_dl <dbl>,
## #   quick_ratio_dl <dbl>, de_ratio_dl <dbl>, debt_assets_dl <dbl>,
## #   intcov_dl <dbl>, adate_year <dbl>, public_year <dbl>, bank_event <date>,
## #   bankruptcy_year <dbl>, Added <int>
```

```
AllBankrupt <- bind_rows(Y12df, Y23df, Y34df)

MasterData <- MasterData %>%
  bind_rows(AllBankrupt) %>%
  arrange(Added, COMPANY_FKEY)
MasterData
```

```
## # A tibble: 101,031 x 91
##    gvkey permno adate      public_date   CAPEI     bm     evm pe_op_basic
##    <int>  <int> <date>     <date>        <dbl>  <dbl>   <dbl>       <dbl>
##  1  1278  77829 2000-12-31 1999-12-31    -0.21  NA      32.4        NA
##  2 63192  83719 1999-12-31 1998-12-31    -2.48   0.9    20.9        -1.34
```

```
##  3  1627  83841 2000-09-30 1999-09-30   -0.037 NA      10.0        -0.036
##  4  2033  29532 2008-09-30 2007-09-30   -1.80    2.03   -2.08       -1.34
##  5  2215  55044 2000-11-30 1999-11-30   61.9    0.357  10.5        10.2
##  6  2393  17961 2020-06-30 2019-06-30  -46.0    3.12   30.3        10.2
##  7  2625  20723 2001-06-30 2000-06-30   -3.22   0.408 -31.5        10.2
##  8  2811  30402 2016-12-31 2015-12-31   -0.109  1.35  -51.1        -0.071
##  9  3156  64901 2008-12-31 2007-12-31   -0.086  1.35   -0.876      -0.056
## 10  2555  20248 2013-01-31 2012-01-31   -1.03   1.35    6.18       -0.219
## # ... with 101,021 more rows, and 83 more variables: pe_op_dil <dbl>,
## #   pe_exi <dbl>, pe_inc <dbl>, ps <dbl>, pcf <dbl>, dpr <dbl>, npm <dbl>,
## #   opmbd <dbl>, opmad <dbl>, gpm <dbl>, ptpm <dbl>, cfm <dbl>, roa <dbl>,
## #   roe <dbl>, roce <dbl>, efftax <dbl>, aftret_eq <dbl>, aftret_invcapx <dbl>,
## #   aftret_equity <dbl>, pretret_noa <dbl>, pretret_earnat <dbl>, GProf <dbl>,
## #   equity_invcap <dbl>, debt_invcap <dbl>, totdebt_invcap <dbl>,
## #   capital_ratio <dbl>, int_debt <dbl>, int_totdebt <dbl>, cash_lt <dbl>,
## #   invt_act <dbl>, rect_act <dbl>, debt_at <dbl>, debt_ebitda <dbl>,
## #   short_debt <dbl>, curr_debt <dbl>, lt_debt <dbl>, profit_lct <dbl>,
## #   ocf_lct <dbl>, cash_debt <dbl>, fcf_ocf <dbl>, lt_ppent <dbl>,
## #   dltt_be <dbl>, debt_assets <dbl>, debt_capital <dbl>, de_ratio <dbl>,
## #   intcov <dbl>, intcov_ratio <dbl>, cash_ratio <dbl>, quick_ratio <dbl>,
## #   curr_ratio <dbl>, cash_conversion <dbl>, inv_turn <dbl>, at_turn <dbl>,
## #   rect_turn <dbl>, pay_turn <dbl>, sale_invcap <dbl>, sale_equity <dbl>,
## #   sale_nwc <dbl>, rd_sale <dbl>, adv_sale <dbl>, staff_sale <dbl>,
## #   accrual <dbl>, ptb <dbl>, PEG_trailing <dbl>, divyield <chr>, TICKER <chr>,
## #   cusip <chr>, fyear <int>, datadate <int>, tic <chr>, COMPANY_FKEY <int>,
## #   cash_debt_dl <dbl>, curr_debt_dl <dbl>, int_totdebt_dl <dbl>,
## #   quick_ratio_dl <dbl>, de_ratio_dl <dbl>, debt_assets_dl <dbl>,
## #   intcov_dl <dbl>, adate_year <dbl>, public_year <dbl>, bank_event <date>,
## #   bankruptcy_year <dbl>, Added <int>
```

```r
Bankrupt <- MasterData %>%
  group_by(COMPANY_FKEY) %>%
  mutate(Added = as.numeric(Added)) %>%
  mutate(Added = if_else(is.na(Added), 0, Added)) %>%
  summarise(Added = max(Added)) %>%
  mutate(isBankrupt = if_else(Added >= 1, 1, 0))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
  sum(Bankrupt$isBankrupt)
```

```
## [1] 474
```

```r
MasterData <- MasterData %>%
  left_join(Bankrupt) %>%
  replace_na(list(isBankrupt = 0))
```

```
## Joining, by = c("COMPANY_FKEY", "Added")
```

```r
library(DescTools)
library(knitr)
```

```r
MasterWinsorized <- MasterData %>%
  mutate(across(-c(gvkey,permno,adate,public_date,adate_year,COMPANY_FKEY,TICKER,fyear,cusip,datadate,t
  mutate(across(-c(gvkey,permno,adate,public_date,adate_year,COMPANY_FKEY,TICKER,fyear,cusip,datadate,t
```

```
## Warning: Problem with ‘mutate()‘ input ‘..1‘.
## i NAs introduced by coercion
## i Input ‘..1‘ is ‘across(...)‘.

## Warning in fn(col, ...): NAs introduced by coercion
```

```r
MasterWinsorized <- as_tibble(MasterWinsorized)

MasterWinsorized  %>%
  filter(isBankrupt == 0) %>%
  summarize(across(c(cash_debt, curr_debt,int_totdebt, quick_ratio, de_ratio, debt_assets, intcov),
                  list(
            mean= ~mean(.),
            sd  = ~sd(.),
            p25 = ~quantile(.,.25),
            p50 = ~median(.),
            p75 = ~quantile(.,.75)),
            na.rm = TRUE
            ))
```

```
## # A tibble: 1 x 35
##   cash_debt_mean cash_debt_sd cash_debt_p25 cash_debt_p50 cash_debt_p75
##            <dbl>        <dbl>         <dbl>         <dbl>         <dbl>
## 1         0.0355        0.600             0         0.116         0.246
## # ... with 30 more variables: curr_debt_mean <dbl>, curr_debt_sd <dbl>,
## #   curr_debt_p25 <dbl>, curr_debt_p50 <dbl>, curr_debt_p75 <dbl>,
## #   int_totdebt_mean <dbl>, int_totdebt_sd <dbl>, int_totdebt_p25 <dbl>,
## #   int_totdebt_p50 <dbl>, int_totdebt_p75 <dbl>, quick_ratio_mean <dbl>,
## #   quick_ratio_sd <dbl>, quick_ratio_p25 <dbl>, quick_ratio_p50 <dbl>,
## #   quick_ratio_p75 <dbl>, de_ratio_mean <dbl>, de_ratio_sd <dbl>,
## #   de_ratio_p25 <dbl>, de_ratio_p50 <dbl>, de_ratio_p75 <dbl>,
## #   debt_assets_mean <dbl>, debt_assets_sd <dbl>, debt_assets_p25 <dbl>,
## #   debt_assets_p50 <dbl>, debt_assets_p75 <dbl>, intcov_mean <dbl>,
## #   intcov_sd <dbl>, intcov_p25 <dbl>, intcov_p50 <dbl>, intcov_p75 <dbl>
```

```r
MasterWinsorized %>%
  filter(isBankrupt==1) %>%
  summarize(across(c(cash_debt_dl, curr_debt_dl,int_totdebt_dl,
                     quick_ratio_dl, de_ratio_dl, debt_assets_dl, intcov_dl),
            list(
            mean=~mean(.),
            sd=~sd(.),
            p25=~quantile(.,.25),
            p50=~median(.),
            p75=~quantile(.,.75)),
            na.rm = TRUE
            ))
```

```
## # A tibble: 1 x 35
##   cash_debt_dl_me~ cash_debt_dl_sd cash_debt_dl_p25 cash_debt_dl_p50
##              <dbl>           <dbl>            <dbl>            <dbl>
## 1           -0.436            4.46            -1.18           -0.504
## # ... with 31 more variables: cash_debt_dl_p75 <dbl>, curr_debt_dl_mean <dbl>,
## #   curr_debt_dl_sd <dbl>, curr_debt_dl_p25 <dbl>, curr_debt_dl_p50 <dbl>,
## #   curr_debt_dl_p75 <dbl>, int_totdebt_dl_mean <dbl>, int_totdebt_dl_sd <dbl>,
## #   int_totdebt_dl_p25 <dbl>, int_totdebt_dl_p50 <dbl>,
## #   int_totdebt_dl_p75 <dbl>, quick_ratio_dl_mean <dbl>,
## #   quick_ratio_dl_sd <dbl>, quick_ratio_dl_p25 <dbl>,
## #   quick_ratio_dl_p50 <dbl>, quick_ratio_dl_p75 <dbl>, de_ratio_dl_mean <dbl>,
## #   de_ratio_dl_sd <dbl>, de_ratio_dl_p25 <dbl>, de_ratio_dl_p50 <dbl>,
## #   de_ratio_dl_p75 <dbl>, debt_assets_dl_mean <dbl>, debt_assets_dl_sd <dbl>,
## #   debt_assets_dl_p25 <dbl>, debt_assets_dl_p50 <dbl>,
## #   debt_assets_dl_p75 <dbl>, intcov_dl_mean <dbl>, intcov_dl_sd <dbl>,
## #   intcov_dl_p25 <dbl>, intcov_dl_p50 <dbl>, intcov_dl_p75 <dbl>
```

```
MasterData %>%
  group_by(isBankrupt,adate_year) %>%
  summarize(n=n()) %>%
  mutate(Prop=n/sum(n)) %>%
  mutate(`Prop_%`=(Prop)*100) %>%
  rename("Bankruptcy (1=Yes)" = "isBankrupt")
```

```
## `summarise()` regrouping output by 'isBankrupt' (override with `.groups` argument)
```

```
## # A tibble: 57 x 5
## # Groups:   Bankruptcy (1=Yes) [2]
##    `Bankruptcy (1=Yes)` adate_year     n   Prop `Prop_%`
##                   <dbl>      <dbl> <int>  <dbl>    <dbl>
##  1                    0       1988  3550 0.0353     3.53
##  2                    0       1989  3552 0.0353     3.53
##  3                    0       1990  3506 0.0349     3.49
##  4                    0       1991  3554 0.0353     3.53
##  5                    0       1992  3639 0.0362     3.62
##  6                    0       1993  3836 0.0381     3.81
##  7                    0       1994  4053 0.0403     4.03
##  8                    0       1995  4182 0.0416     4.16
##  9                    0       1996  4418 0.0439     4.39
## 10                    0       1997  4536 0.0451     4.51
## # ... with 47 more rows
```

```
#Here we're saying that false negatives cost
fn_cost = .4177 #says that false negative cost is 41.77%
fp_cost = .0927 #says that false positive cost is 9.27%

overall_error_function  <- function(r, pi){
    #This is doing false negative rate X bankruptcies X Cost of false negatives
    c1 = (r==1)&(pi<0.5) #logical vector - true if actual 1 but predict 0
    #This is doing false positive rate X non-bankruptcies X Cost of false positives
    c0 = (r==0)&(pi>=0.5) #logical vector - true if actual 0 but predict 1
    return(mean(c1+c0))
```

```r
  }

model_cost_error_function  <- function(r, pi){
    c1 = (r==1)&(pi<0.5) #logical vector - true if actual 1 but predict 0
    c0 = (r==0)&(pi>=0.5) #logical vector - true if actual 0 but predict 1
    return(mean(fn_cost*c1 + fp_cost*c0))
}
```

```r
library(boot)
```

```
##
## Attaching package: 'boot'

## The following object is masked from 'package:qwraps2':
##
##     logit
```

```r
pacman::p_load(caret)

max_poly = 5
poly = 1:max_poly
fold_set = c(5,10)

# Set up storage vectors

v           <-  vector("double",max_poly*length(fold_set))
poly_or_nn  <-  vector("integer",max_poly*length(fold_set))
fold        <-  vector("integer",max_poly*length(fold_set))
estimator   <-  vector("character",max_poly*length(fold_set))
overall_error <- vector("double", max_poly*length(fold_set))
model_error <- vector("double", max_poly*length(fold_set))

ratio_bankrupt <- MasterWinsorized$isBankrupt


for (f in seq_along(fold_set)) {
  for (p in seq_along(poly)) {

    # We want positions 1:5 for folds=5, positions 6:10 for folds=10
    # (f-1)*max_poly+p gets us (1-1)*5+1=1,(1-1)*5+2=2, etc.
    # Once we're at f=2, it gets us (2-1)*5+1=6,(2-1)*5+2=7, etc.

    location = max_poly*(f-1) + p

    logit_fit <-  glm(isBankrupt~
                poly(cash_debt,p) +
                poly(curr_debt,p) +
                poly(int_totdebt,p) +
                poly(quick_ratio,p) +
                poly(de_ratio,p) +
                poly(debt_assets,p) +
                poly(intcov, p),
```

```r
                  data=MasterWinsorized,
               family= "binomial")

    poly_or_nn[location]  <-   p
    fold[location]        <-   fold_set[f]
    estimator[location]   <-   "logit"
    overall_error[location] <- cv.glm(MasterWinsorized, logit_fit, overall_error_function,K = fold_set[
    model_error[location] <- cv.glm(MasterWinsorized, logit_fit, model_cost_error_function,K = fold_set
  }
}

LogitTable <- as_tibble(cbind(estimator,fold,poly_or_nn, overall_error, model_error)) %>%
  mutate(fold=as.integer(fold),
         poly_or_nn=as.integer(poly_or_nn)) %>%
         rename("Estimator"="estimator") %>%
         rename("Fold"="fold") %>%
         rename("Poly_or_Inverse_NN"="poly_or_nn") %>%
         rename("Overall Error"="overall_error") %>%
         rename("Model Error Cost" = "model_error") %>%
         mutate(Poly_or_Inverse_NN=as.double(Poly_or_Inverse_NN)) %>%
         mutate(`Overall Error`=as.double(`Overall Error`)) %>%
         mutate(`Model Error Cost`=as.double(`Model Error Cost`))
LogitTable
```

```
## # A tibble: 10 x 5
##    Estimator  Fold Poly_or_Inverse_NN `Overall Error` `Model Error Cost`
##    <chr>     <int>              <dbl>           <dbl>              <dbl>
##  1 logit         5                  1         0.00471            0.00196
##  2 logit         5                  2         0.00469            0.00196
##  3 logit         5                  3         0.00469            0.00196
##  4 logit         5                  4         0.00469            0.00196
##  5 logit         5                  5         0.00469            0.00196
##  6 logit        10                  1         0.00470            0.00196
##  7 logit        10                  2         0.00469            0.00196
##  8 logit        10                  3         0.00469            0.00196
##  9 logit        10                  4         0.00469            0.00196
## 10 logit        10                  5         0.00469            0.00196
```

KNN SECTION

```r
#Make df
MasterScaled <- MasterData
MasterScaled <- MasterScaled %>% #na.omit() %>%
  mutate(across(c(cash_debt, curr_debt, int_totdebt, quick_ratio, de_ratio, debt_assets, intcov),scale))
  mutate(isBankrupt=ifelse(isBankrupt=="1","B","NB")) %>%
  mutate(isBankrupt=as.factor(isBankrupt))
levels(MasterScaled$isBankrupt)
```

```
## [1] "B"  "NB"
```

```r
MasterScaled %>% select(c(cash_debt, curr_debt, int_totdebt, quick_ratio, de_ratio, debt_assets, intcov
```

```
## # A tibble: 101,031 x 8
##    cash_debt[,1] curr_debt[,1] int_totdebt[,1] quick_ratio[,1] de_ratio[,1]
##            <dbl>         <dbl>           <dbl>           <dbl>        <dbl>
##  1      -0.0139         -0.413        -0.0133           -0.305     -0.0994
##  2       0.0363         -1.17         -0.0121           -0.244      0.0756
##  3       0.0421         -0.861        -0.0120           -0.236     -0.0190
##  4       0.00944         0.243        -0.0105           -0.271      0.00161
##  5      -0.0466          0.989        -0.0141           -0.139      0.00459
##  6      -0.0320         -0.193        -0.0154           -0.274      0.00533
##  7      -0.609           1.77         -0.0164            0.201     -0.00649
##  8      -0.0419         -1.11         -0.0112           -0.137      0.114
##  9      -0.537           0.362        -0.00360          -0.207     -0.0439
## 10      -0.0349          0.884        -0.0152           -0.337     -0.0173
## # ... with 101,021 more rows, and 3 more variables: debt_assets[,1] <dbl>,
## #   intcov[,1] <dbl>, isBankrupt <fct>
```

```
pacman::p_load(caret)

max_nn=5
nn_set=(1:max_nn)
fold_set_knn=c(5,10)

ratio_bankrupt <- mean(MasterWinsorized$isBankrupt == 1)
ratio_non_bankrupt <- 1 - ratio_bankrupt

# Set up storage vectors

v             <-  vector("double",max_nn*length(fold_set_knn))
poly_or_nn    <-  vector("integer",max_nn*length(fold_set_knn))
fold          <-  vector("integer",max_nn*length(fold_set_knn))
estimator     <-  vector("character",max_nn*length(fold_set_knn))
expected_error_costs_knn  <- vector("double", max_nn*length(fold_set_knn))
overall_error_knn         <- vector("double", max_nn*length(fold_set_knn))

set.seed(1313)

for (f in seq_along(fold_set_knn)) {

#For overall error rate
trControl_fit1 <- trainControl(method  = "cv",
                            number  = fold_set_knn[f],
                            preProcOptions=c("scale"))


  #This fits the model, for different numbers of near neighbors
fit1 <- train(isBankrupt ~ cash_debt+curr_debt+int_totdebt+quick_ratio+de_ratio+debt_assets+intcov,
               method     = "knn",
               tuneGrid   = expand.grid(k = 1:max_nn),
               trControl  = trControl_fit1,
               metric     = "Accuracy",
               data       = MasterScaled)

#For Sensitivity and Specificity
trControl_knn_fit <- trainControl(method  = "cv",
```

```
                              number  = fold_set_knn[f],
                              classProbs=TRUE,
                              summaryFunction = twoClassSummary,
                              preProcOptions=c("scale"))

knn_fit <- train(isBankrupt ~ cash_debt+curr_debt+int_totdebt+quick_ratio+de_ratio+debt_assets+intcov,
                 method     = "knn",
                 tuneGrid   = expand.grid(k = 1:max_nn),
                 trControl  = trControl_knn_fit,
                 data       = MasterScaled)

  #you want the first subset to lie between index 1 and 5, the second from 6 and 10, for the first set,
  #for the first set,  max=5*(1-1)+5=5, for the second set,  min=5*(2-1)+1=6, for the second set,  max=

  min=max_nn*(f-1)+1
  max=max_nn*(f-1)+max_nn


  poly_or_nn[min:max]   <-  1/nn_set
  fold[min:max]         <-  fold_set_knn[f]
  estimator[min:max]    <-  "knn"
  expected_error_costs_knn[min:max] <- (((fn_cost)*ratio_bankrupt*(1-knn_fit$results$Sens))+((fp_cost)*
  overall_error_knn[min:max] <- (1-fit1$results$Accuracy)
  }
```

```
## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was not
## in the result set. ROC will be used instead.
```

```
KNNTable <- as_tibble(cbind(estimator,fold_set_knn,poly_or_nn,overall_error_knn,expected_error_costs_kn
         mutate(poly_or_nn=round(as.numeric(poly_or_nn),2)) %>% mutate(overall_error_knn = as.numeric(o
          mutate(expected_error_costs_knn = as.numeric(expected_error_costs_knn)) %>%
        rename("Poly_or_Inverse_NN"="poly_or_nn") %>% rename("Model Error Cost"="expected_error_costs_k
         rename("Overall Error"="overall_error_knn") %>% rename("Estimator"="estimator") %>%
        rename("Fold"="fold_set_knn")
KNNTable
```

```
## # A tibble: 10 x 5
##    Estimator  Fold Poly_or_Inverse_NN `Overall Error` `Model Error Cost`
##    <chr>     <dbl>              <dbl>           <dbl>              <dbl>
##  1 knn           5               1            0.0100             0.00250
##  2 knn           5               0.5          0.00890            0.00235
##  3 knn           5               0.33         0.00478            0.00197
##  4 knn           5               0.25         0.00477            0.00196
##  5 knn           5               0.2          0.00469            0.00196
##  6 knn          10               1            0.0103             0.00248
##  7 knn          10               0.5          0.00887            0.00233
##  8 knn          10               0.33         0.00478            0.00196
##  9 knn          10               0.25         0.00474            0.00196
## 10 knn          10               0.2          0.00470            0.00196
```

```r
TotalTable <- bind_rows(LogitTable,KNNTable)
TotalTable
```

```
## # A tibble: 20 x 5
##    Estimator  Fold Poly_or_Inverse_NN `Overall Error` `Model Error Cost`
##    <chr>     <dbl>              <dbl>           <dbl>              <dbl>
##  1 logit         5               1            0.00471            0.00196
##  2 logit         5               2            0.00469            0.00196
##  3 logit         5               3            0.00469            0.00196
##  4 logit         5               4            0.00469            0.00196
##  5 logit         5               5            0.00469            0.00196
##  6 logit        10               1            0.00470            0.00196
##  7 logit        10               2            0.00469            0.00196
##  8 logit        10               3            0.00469            0.00196
##  9 logit        10               4            0.00469            0.00196
## 10 logit        10               5            0.00469            0.00196
## 11 knn           5               1            0.0100             0.00250
## 12 knn           5               0.5          0.00890            0.00235
## 13 knn           5               0.33         0.00478            0.00197
## 14 knn           5               0.25         0.00477            0.00196
## 15 knn           5               0.2          0.00469            0.00196
## 16 knn          10               1            0.0103             0.00248
## 17 knn          10               0.5          0.00887            0.00233
## 18 knn          10               0.33         0.00478            0.00196
## 19 knn          10               0.25         0.00474            0.00196
## 20 knn          10               0.2          0.00470            0.00196
```

According to the table, which includes overall error and model error costs for both the logit and KNN models, the overall error for the logit model is lower than that of the KNN model, but overall errors for both models are relatively similar and are within .00469 to .01029. The optimal logit model appears to have a polynomial term greater than 1 in 10-fold cross validation, which minimizes the overall error for 10-fold cross validation. The overall error for 5-fold cross validation was constant regardless of the polynomial term. The optimal KNN model appears to be K=5 near neighbors in 5-fold cross validation, which minimizes the overall error rate while having a lower model error cost than the second-best KNN model, which was K=5 near neighbors in 10-fold cross validation. The model error costs for both the logit and KNN models are also relatively similar and are within .001959 to .0025.